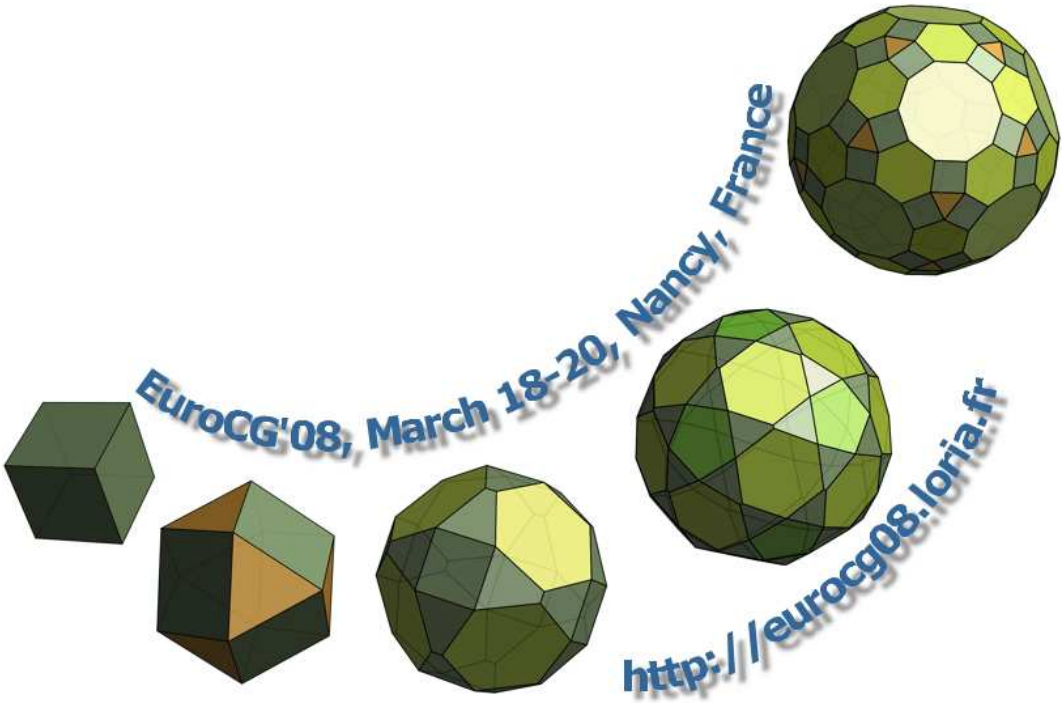
Collection of abstracts of the 24th European Workshop on Computational Geometry

Sylvain Petitjean

Collection of Abstracts



supported by

INRIA
Nancy-Université
lorraine conseil régional
GDR IM
Grand Nancy communauté urbaine
ifp
Meurthe & Moselle Conseil Général
inpl nancy
Loria
Dassault Systèmes

Collection of abstracts of the
24th European Workshop on Computational Geometry
LORIA, Nancy, France
March 18-20, 2008

For information about obtaining copies of this volume, contact

Sylvain Petitjean
LORIA
Campus scientifique
BP 239
54506 Vandœuvre cedex
France

Sylvain.Petitjean@loria.fr

Cover art by Craig S. Kaplan, University of Waterloo.
Panoramic picture of Place Stanislas by Emmanuel Faivre.
Cover design by Sophie Nertomb and Sylvain Petitjean.

Preface

The **24th European Workshop on Computational Geomety** (EuroCG'08) was held at the Laboratoire Lorrain de Recherche en Informatique et ses Applications (LORIA) on March 18-20, 2008. It was preceded by a one-day workshop entitled "CGAL Innovations and Applications: Robust Geometric Software for Complex Shapes" held on March 17, 2008. More information about both events can be found at `http://eurocg08.loria.fr` (see also `http://www.eurocg.org` for previous workshops).

The present collection of abstracts contains the 63 scientific contributions as well as three invited talks presented at the workshop. It is also available electronically from the workshop's web site at `http://eurocg08.loria.fr/EuroCG08Abstracts.pdf`. This year's record of 72 submissions with authors from 22 different countries, covering a wide range of topics, shows that Computational Geometry is a lively and still growing research field in Europe.

Following the tradition of the workshop, many contributions present ongoing research, and it is expected that most of them will appear in a more complete version in scientific journals. Selected papers from the workshop will be invited to a special issue of *Computational Geometry: Theory and Applications*. We thank the editors-in-chief, Kurt Mehlhorn and Jörg-Rüdiger Sack, for their cooperation.

We would also like to thank all the authors for submitting papers and presenting their results at the workshop. We are especially grateful to our keynote speakers Pierre Alliez, Jean Ponce, and Fabrice Rouillier for accepting our invitation. Special thanks go to our sub-referees and to Bettina Speckmann for providing us with the LaTeX class used to format this collection.

Finally, we are grateful to LORIA for providing the necessary infrastructure, and to our sponsors for their support: INRIA, Université Henri Poincaré, Université Nancy 2, Institut National Polytechnique de Lorraine, GDR Informatique Mathématique of CNRS, Communauté Urbaine du Grand Nancy, Conseil Général de Meurthe-et-Moselle, Conseil Régional de Lorraine, Dassault Systèmes and Institut Français du Pétrole.

Note that the next edition of EuroCG will be held in 2009 in Brussels, Belgium.

Sylvain Petitjean (Editor)

Table of Contents

Tuesday, March 18

9:30 - 10:30, Session 1

10:50 - 11:50, Session 2

12:00 - 13:00, Invited Talk 1

14:30 - 15:50, Session 3A

14:30 - 15:50, Session 3B

16:10 - 17:30, Session 4A

16:10 - 17:30, Session 4B

Wednesday, March 19

9:30 - 10:30, Session 5

10:50 - 11:50, Session 6

12:00 - 13:00, Invited Talk 2

14:30 - 15:50, Session 7A

14:30 - 15:50, Session 7B

16:10 - 17:30, Session 8A

16:10 - 17:30, Session 8B

Thursday, March 20

9:30 - 10:30, Session 9A

9:30 - 10:30, Session 9B

10:50 - 11:50, Session 10A

10:50 - 11:50, Session 10B

12:00 - 13:00, Invited Talk 3

14:30 - 15:30, Session 11

Delaunay Edge Flips in Dense Surface Triangulations*

Siu-Wing Cheng† Tamal K. Dey‡

Abstract

We study the conversion of a surface triangulation to a subcomplex of the Delaunay triangulation with edge flips. We show that the surface triangulations which closely approximate a smooth surface with uniform density can be transformed to a Delaunay triangulation with a simple edge flip algorithm. The condition on uniformity becomes less stringent with increasing density of the triangulation. If the condition is dropped, the output surface triangulation becomes "almost Delaunay" instead of exactly Delaunay.

1 Introduction

The importance of computing Delaunay triangulations in applications of science and engineering cannot be overemphasized. Among the different Delaunay triangulation algorithms, flip based algorithms are most popular and perhaps the most dominant approach in practice. The sheer elegance and simplicity of this approach make it attractive to implement.

In $\mathbb{R}^2$, if the circumcircle of a triangle t contains a vertex of another triangle t' sharing an edge e with it, *flipping* e means replacing e with the other diagonal edge contained in the union of t and t'. A well-known elegant result is that this process terminates and produces the Delaunay triangulation. In higher dimensions, the edge flips can be naturally extended to bistellar flips. Edelsbrunner and Shah [8] showed that bi-stellar flips can be used with incremental point insertion to construct weighted Delaunay triangulations in three and higher dimensions.

Given the increasing demand of computing surface triangulations that are sub-complexes of Delaunay triangulations [1, 6, 7], it is natural to ask if a surface triangulation can be converted to a Delaunay one by edge flips and, if so, under what conditions. Once the surface triangulation is made Delaunay, a number of tools that exploit Delaunay properties can be used for further processing.

We show that a dense triangulation can be flipped to a Delaunay triangulation if the density is uniform in some sense. The practical implication of this result is that reasonably dense triangulations can be converted to Delaunay triangulations with a simple edge flip algorithm. Furthermore, the results in this paper have been used for a recent algorithm on maintaining deforming meshes with provable guarantees [5]. What happens if we do not have the uniformity condition? We show that the flip algorithm still terminates but the output surface may not be Delaunay. Nonetheless, this surface is "almost Delaunay" in the sense that the diametric ball of each triangle shrunk by a small amount remains empty. These approximate Delaunay triangulations may find applications where exact Delaunay triangulations are not required; for example, see the work by Bandyopadhyay and Snoeyink [3].

2 Preliminaries

Surface. Let $\Sigma \subset \mathbb{R}^3$ be a smooth compact surface without boundary. The *medial axis* is the set of centers of all maximally empty balls. The *reach* γ of Σ is the infimum over Euclidean distances of all points in Σ to its medial axis. Let $\mathbf{n}_x$ denote the outward unit normal of Σ at a point $x \in \Sigma$.

Triangulation. We say T is a triangulation of a surface Σ if vertices of T lie in Σ and its underlying space $|T|$ is homeomorphic to Σ. For any triangle $t \in T$, $\mathbf{n}_t$ denotes the outward unit normal of t.

The triangulation T has a *consistent orientation* if for any triangle $t \in T$ and for any vertex q of t, $\angle \mathbf{n}_t, \mathbf{n}_q \leq \frac{\pi}{2}$.

If a triangle $t \in T$ shares an edge pq with a triangle pqs, we call s a *neighbor* vertex of t. Let $\rho(t)$ denote the circumradius of t. The ratio of $\rho(t)$ to the shortest edge length of t is called the *radius-edge ratio*. We call the maximum radius-edge ratio of triangles in T the radius-edge ratio of T.

We call T ε-dense for some $\varepsilon < 1$ if $\rho(t) \leq \varepsilon\gamma$ for each triangle $t \in T$ and T has a consistent orientation. Also, if the distance between any two vertices in T is at least $\delta\varepsilon\gamma$ for some $\delta < 1$, we call T (ε, δ)-dense.

Stab and flip. Let $B(c, r)$ denote the ball with center c and radius r. A *circumscribing ball* of a triangle t is any ball that has the vertices of t on its boundary. The *diametric ball* is the smallest such ball and we denote it by D_t.

*Research supported by NSF grants CCF-0430735 and CCF-0635008 and Research Grant Council, Hong Kong, China (612107).

†Department of Computer Science and Engineering, HKUST, Clear Water Bay, Hong Kong, scheng@cse.ust.hk

‡Department of Computer Science and Engineering, The Ohio State University, Columbus, OH 43210, USA, tamaldey@cse.ohio-state.edu

A vertex v of T *stabs* a ball B if v lies inside B. A triangle $t \in T$ is *stabbed* if D_t is stabbed by a vertex of T. The triangle t is *locally stabbed* if the stabbing vertex is one of the three neighbor vertices of t.

The common edge pq between two triangles pqr and pqs in T is *flippable* if pqr is stabbed by s (i.e., locally stabbed). We will show later that this definition is symmetric. Flipping pq means replacing pqr and pqs by the triangles prs and qrs. If the new triangulation is T' we write $T \xrightarrow{pq} T'$.

Power distance. Given a point x and a ball $B(c,r)$, the *power distance* $\text{pow}(x, B(c,r))$ is $\|c-x\|^2 - r^2$. Given two balls B_1 and B_2, their *bisector* $C(B_1, B_2)$ consists of points at equal power distances from B_1 and B_2. It turns out that $C(B_1, B_2)$ is a plane. If B_1 and B_2 intersect, $C(B_1, B_2)$ contains the circle $\partial B_1 \cap \partial B_2$.

Background results. The following previous results on normal approximations will be useful.

Lemma 1 ([2, 4]) *For any two points x and y in Σ such that $\|x-y\| \leq \varepsilon\gamma$ for some $\varepsilon \leq \frac{1}{3}$, $\angle \mathbf{n}_x, \mathbf{n}_y \leq \frac{\varepsilon}{1-\varepsilon}$ and $\angle \mathbf{n}_x, (y-x) \geq \arccos(\frac{\varepsilon}{2})$.*

Combining Lemmas 1 and a result in [7], we get:

Corollary 2 *Let T be an ε-dense triangulation for some $\varepsilon < 0.1$. For any vertex q of a triangle $t \in T$, $\angle \mathbf{n}_t, \mathbf{n}_q \leq 7\varepsilon$.*

Define the *dihedral angle* between two adjacent triangles pqr and qrs as $\angle \mathbf{n}_{pqr}, \mathbf{n}_{qrs}$. Corollary 2 implies that:

Corollary 3 *Let T be an ε-dense triangulation for some $\varepsilon < 0.1$. For any two adjacent triangles pqr and qrs in T, $\angle \mathbf{n}_{pqr}, \mathbf{n}_{qrs} \leq 14\varepsilon$.*

3 Flip algorithm

The flip algorithm that we consider is very simple: as long as there is a flippable edge, flip it.

MeshFlip(T)

1. If there is a flippable edge $e \in T$ then flip e else output T;
2. $T := T'$ where $T \xrightarrow{e} T'$; go to step 1.

There are two issues. First, under what condition does MeshFlip terminate? Second, what triangulation does MeshFlip produce? In this section, we show that MeshFlip terminates if T is an ε-dense triangulation. We address the second issue later.

The following lemma establishes the symmetry in local stabbing.

Lemma 4 *Let pqr and pqs be two adjacent triangles such that s stabs pqr. If $\angle \mathbf{n}_{pqr}, \mathbf{n}_{pqs} < \frac{\pi}{2}$, r stabs pqs.*

Proof. It can be shown that the bisector C_{pq} separates r and s if pqr and pqs make an angle larger than $\frac{\pi}{2}$ or equivalently $\angle \mathbf{n}_{pqr}, \mathbf{n}_{pqs} < \frac{\pi}{2}$. Let C^+_{pq} be the half-space supported by C_{pq} and containing s. Define C^-_{pq} similarly as the half-space not containing s. Clearly, $D_{pqs} \cap C^+_{pq} \subset D_{pqr} \cap C^+_{pq}$ as s is on the boundary of D_{pqs}. So $D_{pqr} \cap C^-_{pq} \subset D_{pqs} \cap C^-_{pq}$. But C^-_{pq} contains r which is on the boundary of D_{pqr}. So r is inside D_{pqs}. □

Next, we show that an edge flip produces two triangles with a smaller maximum circumradius.

Lemma 5 *Let T be a triangulation with dihedral angles less than $\frac{\pi}{2}$. Let $pqr, pqs \in T$ be triangles such that s stabs pqr. Then $\rho(qrs) \leq \max\{\rho(pqr), \rho(pqs)\}$ and $\rho(prs) \leq \max\{\rho(pqr), \rho(pqs)\}$.*

Proof. We prove the lemma for $\rho(qrs)$. The analysis for $\rho(prs)$ is similar. Consider the bisectors $C_{qr} = C(D_{pqr}, D_{qrs})$ and $C_{qs} = C(D_{pqs}, D_{qrs})$. Let C^+_{qr} be the half-space supported by C_{qr} containing s. Let C^+_{qs} be the half-space supported by C_{qs} containing p.

By assumption the dihedral angle between pqr and pqs is at most $\frac{\pi}{2}$. Then, Lemma 4 applies to claim that r stabs pqs.

Clearly, the center of D_{qrs} lies in the union $C^+_{qr} \cup C^+_{qs}$. First, assume that C^+_{qr} contains the center of D_{qrs}. Clearly, $D_{qrs} \cap C^+_{qr} \subset D_{pqr} \cap C^+_{qr}$ as s is contained in D_{pqr} by the assumption that s stabs pqr. This implies that $D_{pqr} \cap C^+_{qr}$ contains the center of D_{qrs}. So D_{qrs} is smaller than D_{pqr} establishing the claim. If C^+_{qs} contains the center of D_{qrs}, the above argument can be repeated by replacing C^+_{qr} with C^+_{qs}, D_{pqr} with D_{pqs}, and s with r. □

Since the maximum circumradius decreases monotonically by the edge flips, the triangles can still be oriented consistently with Σ and a homeomorphism using closest point map [7] can be established between Σ and the new triangulation. Hence, the new triangulation satisfies the conditions for being ε-dense.

Corollary 6 *If $T \xrightarrow{e} T'$ for a flippable edge e and T is ε-dense for some $\varepsilon < 1$, then T' is also ε-dense.*

Lemma 7 *If T is ε-dense for some $\varepsilon < 0.1$, then* MeshFlip(T) *terminates.*

Proof. Let $R_1, R_2, .., R_n$ be the decreasing sequence of the radii of the diametric balls of the triangles at any instant of the flip process. First of all, an edge flip preserves the number of triangles in the triangulation. An edge flip may change the entries in this sequence

of radii, but not its length. We claim that after a flip the new radii sequence $R'_1, R'_2, ..., R'_n$ decreases lexicographically, that is, there is a j such that $R_i = R'_i$ for all $1 \leq i \leq j$ and $R_{j+1} > R'_{j+1}$. Let $j+1$ be the first index where $R_{j+1} \neq R'_{j+1}$. Since each flip maintains ε-density (Corollary 6), the dihedral angles between adjacent triangles remain at most 14ε by Corollary 3. This angle is less than $\frac{\pi}{2}$ for $\varepsilon < 0.1$. One can apply Lemma 5 to each intermediate triangulation. By this lemma, the maximum of the two radii before a flip decreases after the flip. It means that the triangle corresponding to the radius R_{j+1} has been flipped and its place has been taken by a triangle whose circumradius is smaller than R_{j+1}. So the new radii sequence is smaller lexicographically. It follows that the same triangulation cannot appear twice during the flip sequence. As there are finitely many possible triangulations, MeshFlip must terminate. □

4 Uniform dense triangulation

We prove that MeshFlip can turn an (ε, δ)-dense triangulation to a Gabriel triangulation where no diametric ball of any triangle is stabbed.

Lemma 8 *Assume that a vertex v stabs a triangle pqr in an ε-dense triangulation for some $\varepsilon < 0.1$. Let $\bar{v}$ be the point in pqr closest to v. The angle between $v\bar{v}$ and the support line of $\mathbf{n}_{pqr}$ is at least $\frac{\pi}{2} - 26\varepsilon$.*

Proof. Let T be an ε-dense triangulation of a surface Σ with reach γ. Since v stabs D_{pqr}, we have $\|p - v\| \leq 2\varepsilon\gamma$ which implies that $\|v - \bar{v}\| \leq 2\varepsilon\gamma$. Walk from v towards $\bar{v}$ and let abc be the first triangle in T that we hit. Let y be the point in abc that we hit. (The triangle abc could possibly be pqr.) We have $\|v - y\| \leq \|v - \bar{v}\| \leq 2\varepsilon\gamma$. By the ε-density assumption, we have $\|a - y\| \leq 2\varepsilon\gamma$. It follows that $\|a - v\| \leq \|a - y\| + \|v - y\| \leq 4\varepsilon\gamma$. Then, $\angle\mathbf{n}_v, \mathbf{n}_a < 8\varepsilon$ by Lemma 1, and $\angle\mathbf{n}_{abc}, \mathbf{n}_a \leq 7\varepsilon$ by Corollary 2. Therefore, $\angle\mathbf{n}_v, \mathbf{n}_{abc} < 8\varepsilon + 7\varepsilon = 15\varepsilon$.

Let ℓ be an oriented line through v and $\bar{v}$ such that ℓ enters the polyhedron bounded by T at $y \in abc$ and then exits at v. Assume to the contrary that ℓ makes an angle less than $\frac{\pi}{2} - 26\varepsilon$ with $\mathbf{n}_{pqr}$. Since $\|p - v\| \leq 2\varepsilon\gamma$, Lemma 1 and Corollary 2 imply that $\angle\mathbf{n}_v, \mathbf{n}_{pqr} \leq 4\varepsilon + 7\varepsilon = 11\varepsilon$. Thus, ℓ makes an angle less than $\frac{\pi}{2} - 15\varepsilon$ with $\mathbf{n}_v$. Since $\angle\mathbf{n}_v, \mathbf{n}_{abc} < 15\varepsilon$, ℓ must make an angle less than $\frac{\pi}{2}$ with $\mathbf{n}_{abc}$. Because ℓ enters at y and then exits at v, $\angle\mathbf{n}_v, \mathbf{n}_{abc}$ is greater than $\pi - (\frac{\pi}{2} - 15\varepsilon) - \frac{\pi}{2} = 15\varepsilon$, contradicting the previous deduction that $\angle\mathbf{n}_v, \mathbf{n}_{abc} < 15\varepsilon$. □

Lemma 9 *Assume that a vertex v stabs a triangle pqr in an ε-dense triangulation for some $\varepsilon < 0.1$. There exists an edge, say pq, such that r and v are separated by the plane H_{pq} that contains pq and is perpendicular to pqr.*

Proof. By Lemma 8, $v\bar{v}$ makes a positive angle with the line of $\mathbf{n}_{pqr}$. It follows that v does not project orthogonally onto a point inside pqr. Hence, there exists an edge pq such that H_{pq} separates r and v. □

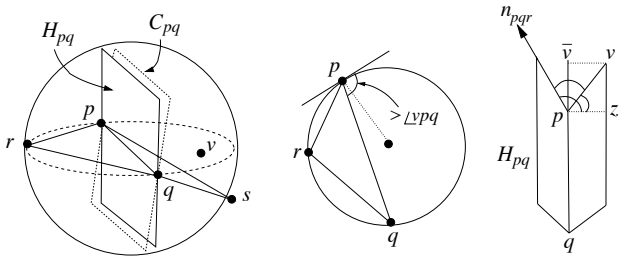


Figure 1: (left) : triangle pqr is stabbed by v. Both v and s lie on the same side of H_{pq} and C_{pq}. The case of v being in the thin wedge between H_{pq} and C_{pq} is eliminated if pqr has bounded radius-edge ratio. (middle) : the worst case for angle $\angle vpq$. (right): the planes of H_{pq} and vpq make large angle ensuring v and s are on the same side of C_{pq}.

Lemma 10 *Assume that a vertex v stabs a triangle pqr in an ε-dense triangulation with radius-edge ratio $a < \frac{1}{2\sin 24\varepsilon}$. If $\varepsilon < \frac{\pi}{72}$, pqr is locally stabbed or v stabs a triangle t such that $\mathrm{pow}(v, D_t) < \mathrm{pow}(v, D_{pqr})$.*

Proof. By Lemma 9, there is a plane H_{pq} through the edge pq and perpendicular to pqr such that H_{pq} separates r and v. Let pqs be the other triangle incident to pq. If s lies inside D_{pqr}, pqr is locally stabbed and we are done. So assume that s does not lie inside D_{pqr}. By Corollary 3, $\angle\mathbf{n}_{pqr}, \mathbf{n}_{pqs} \leq 14\varepsilon$, which is less than $\frac{\pi}{2}$ for $\varepsilon < \frac{\pi}{72}$. Therefore, H_{pq} separates r and s too. It means that v and s lie on the same side of H_{pq}; see Figure 1.

Let C_{pq} denote the bisector $C(D_{pqr}, D_{pqs})$. Let C^+_{pq} be the half-space bound by C_{pq} containing s. It follows that $D_{pqr} \cap C^+_{pq} \subset D_{pqs} \cap C^+_{pq}$ as s lies outside D_{pqr}. Suppose that C^+_{pq} contains v. Then, v lies inside D_{pqs} as v lies inside D_{pqr}. This immediately implies that v stabs pqs and $\mathrm{pow}(v, D_{pqs}) < \mathrm{pow}(v, D_{pqr})$. Therefore, the lemma holds if we can show that C^+_{pq} contains v. This is exactly where we need bounded aspect ratios for triangles.

Let $\bar{s}$ and $\bar{v}$ be the orthogonal projections of s and v respectively onto the line of pq. Consider the following facts.

(i) The acute angle between $s\bar{s}$ and $\mathbf{n}_{pqr}$ is equal to $\frac{\pi}{2} - \angle\mathbf{n}_{pqr}, \mathbf{n}_{pqs}$, which is at least $\frac{\pi}{2} - 14\varepsilon$ by Corollary 3.

(ii) The angle between H_{pq} and C_{pq} cannot be larger than $\angle\mathbf{n}_{pqr}, \mathbf{n}_{pqs}$ which is at most 14ε.

(iii) We prove that $\angle \mathbf{n}_{pqr}, v\bar{v} > \angle \mathbf{n}_{pqr}, \mathbf{n}_{pqs} \geq \angle H_{pq}, C_{pq}$.

These facts imply that v and s on the same side of C_{pq} as H_{pq}. Therefore, it suffices to prove (iii).

First, observe that if $\bar{v}$ is the closest point of v in pq, we have by Lemma 8

$$\angle \mathbf{n}_{pqr}, v\bar{v} \geq \frac{\pi}{2} - 26\varepsilon \geq 14\varepsilon \geq \angle \mathbf{n}_{pqr}, \mathbf{n}_{pqs}.$$

So, assume the contrary. In that case, the closest point of v in pq is either p or q. Assume it to be p. Since $\bar{v}$ lies outside pq, the angle $\angle vpq$ is obtuse. We claim that this angle cannot be arbitrarily close to π. In fact, this angle cannot be more than the maximum obtuse angle pq makes with the tangent plane of D_{pqr} at p. Simple calculation (Figure 1(middle)) shows that this angle is $\frac{\pi}{2} + \arccos \|p-q\|/2\rho(pqr)$ giving

$$\angle vpq \leq \frac{\pi}{2} + \arccos \frac{1}{2a}.$$

Since D_{pqr} contains v, $\|v-p\| \leq 2\varepsilon\gamma$. By Lemma 1, $\angle \mathbf{n}_p, vp \geq \arccos \varepsilon$. Applying Corollary 2, we get

$$\angle \mathbf{n}_{pqr}, vp \geq \angle \mathbf{n}_p, vp - \angle \mathbf{n}_{pqr}, \mathbf{n}_p \geq \arccos \varepsilon - 7\varepsilon.$$

Let $zp \parallel v\bar{v}$ (Figure 1(right)). Then, $\angle v\bar{v}, vp = \angle vpz = \angle vpq - \frac{\pi}{2} \leq \arccos \frac{1}{2a}$. One has $\angle \mathbf{n}_{pqr}, v\bar{v} \geq \angle \mathbf{n}_{pqr}, vp - \angle v\bar{v}, vp = \angle \mathbf{n}_{pqr}, vp - \angle vpz \geq \arccos \varepsilon - 7\varepsilon - \arccos \frac{1}{2a} \geq \frac{\pi}{2} - 10\varepsilon - \arccos \frac{1}{2a}$ for $\varepsilon < \frac{\pi}{72}$. We are now left to show that $\frac{\pi}{2} - 10\varepsilon - \arccos \frac{1}{2a} > \angle \mathbf{n}_{pqr}, \mathbf{n}_{pqs}$ which requires $\frac{\pi}{2} - 24\varepsilon > \arccos \frac{1}{2a}$ or $a < \frac{1}{2 \sin 24\varepsilon}$. This is precisely the condition required by the lemma. □

We are ready to prove the main results of this section.

Theorem 11 *For any $\varepsilon < \frac{\pi}{72}$ and $\delta = 2 \sin 24\varepsilon$, an (ε, δ)-dense triangulation has a stabbed triangle if and only if it has a locally stabbed triangle.*

Proof. The 'if' part is obvious. Consider the 'only if' part. Let pqr be stabbed by v. As $\delta = 2 \sin 24\varepsilon$, the radius-edge ratio is at most $1/(2 \sin 24\varepsilon)$. By Lemma 10, pqr is locally stabbed or v stabs a triangle t where $\text{pow}(v, D_t) < \text{pow}(v, D_{pqr})$. In the latter case, repeat the argument with t. We must reach a locally stabbed triangle since the power distance of v from the diametric balls cannot decrease indefinitely. □

Theorem 12 *For any $\varepsilon < \frac{\pi}{72}$ and $\delta = 2 \sin 24\varepsilon$, an (ε, δ)-dense triangulation can be flipped to a Gabriel triangulation.*

Proof. The maximum circumradius decreases after each flip and the nearest neighbor distance cannot be decreased by flips. So MeshFlip maintains the (ε, δ)-dense conditions after each flip. By Theorem 11, all triangles are Gabriel upon termination. □

5 Dense triangulations

We also study the effect of MeshFlip on an ε-dense triangulation T without the uniformity condition.

Take a triangle $t \in T$. Let c be its circumcenter. A *β-ball* of t is a circumscribing ball centered at $c + \beta \mathbf{n}_t$. The triangle t is *β-stabbed* if a vertex stabs the β-ball and $(-\beta)$-ball of t. We call t *locally β-stabbed* if the stabbing vertex is one of the three neighbor vertices of t.

If we decrease the radius of D_t by β, we get a smaller concentric ball which we denote by D_t^β. We call T β-Gabriel if for each triangle $t \in T$, D_t^β is not stabbed by any vertex of T.

Theorem 13 *For any $\varepsilon < 0.1$, an ε-dense triangulation of a surface with reach γ contains a β-stabbed triangle only if it contains a locally $(\beta - 88\varepsilon^2\gamma)$-stabbed triangle.*

By choosing $\beta = 88\varepsilon^2\gamma$, Theorem 13 implies that no triangle is $88\varepsilon^2\gamma$-stabbed at the termination of MeshFlip. So the output triangulation is $88\varepsilon^2\gamma$-Gabriel.

Theorem 14 *For any $\varepsilon < 0.1$, an ε-dense triangulation of a surface with reach γ can be flipped to a $88\varepsilon^2\gamma$-Gabriel triangulation.*

The omitted details can be found in the full version available at the authors' webpages.

References

[1] N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. *Discr. Comput. Geom.*, 22 (1999), 481–504.

[2] N. Amenta and T. K. Dey. Normal variation for adaptive feature size. http://www.cse.ohio-state.edu/∼tamaldey/paper/norvar/norvar.pdf.

[3] D. Bandyopadhyay and J. Snoeyink. Almost-Delaunay simplices : nearest neighbor relations for imprecise points. *Proc. 15th ACM-SIAM Sympos. Disc. Alg.*, 2004, 410–419.

[4] H.-L. Cheng, T. K. Dey, H. Edelsbrunner and J. Sullivan. Dynamic skin triangulation. *Discr. Comput. Geom.* , 25 (2001), 525–568.

[5] S.-W. Cheng and T. K. Dey. Maintaining deforming surface meshes. *Proc. 19th ACM-SIAM Sympos. Discr. Alg.*, 2008, 112-121.

[6] S.-W. Cheng, T. K. Dey, E. A. Ramos, and T. Ray. Sampling and meshing a surface with guaranteed topology and geometry. *SIAM J. Computing*, 37 (2007), 1199–1227.

[7] T. K. Dey. Curve and surface reconstruction : Algorithms with mathematical analysis. Cambridge University Press, New York, 2006.

[8] H. Edelsbrunner and N. R. Shah. Incremental topological flipping works for regular triangulations. *Algorithmica*, 15 (1996), 223–241.

Decomposing Non-Convex Fat Polyhedra

Mark de Berg* Chris Gray*

Abstract

We show that any locally-fat polyhedron with n vertices and convex fat faces can be decomposed into $O(n)$ tetrahedra. We also show that the additional restriction that the faces are fat is necessary: there are fat polyhedra without fat faces that require $\Omega(n^2)$ pieces in any convex decomposition. Finally, we show that if we want the tetrahedra in the decomposition to be fat themselves, then the number of tetrahedra cannot be bounded as a function of n.

1 Introduction

Polyhedra and their planar equivalent, polygons, play an important role in many geometric problems. From an algorithmic point of view, however, general polyhedra are unwieldy to handle directly: several algorithms can only handle *convex* polyhedra, preferably of *constant complexity*. Hence, there has been extensive research into decomposing polyhedra into tetrahedra or other constant-complexity convex pieces. The two main issues in developing decomposition algorithms are (i) to keep the number of pieces in the decomposition small, and (ii) to compute the decomposition quickly.

In the planar setting the number of pieces is, in fact, not an issue if the pieces should be triangles: any polygon admits a triangulation, and any triangulation of a polygon with n vertices has $n-2$ triangles. Hence, research focused on developing fast triangulation algorithms, culminating in Chazelle's linear-time triangulation algorithm [7]. An extensive survey of algorithms for decomposing polygons and their applications is given by Keil [10].

For 3-dimensional polyhedra, however, the situation is much less rosy. First of all, not every non-convex polyhedron admits a tetrahedralization: there are polyhedra that cannot be decomposed into tetrahedra without using Steiner points. Moreover, deciding whether a polyhedron admits a tetrahedralization without Steiner points is NP-complete [12]. Thus we have to settle for decompositions using Steiner points. Chazelle [6] has shown that any polyhedron with n vertices can be decomposed into $O(n^2)$ tetrahedra, and that this is tight in the worst case: there are polyhedra with n vertices for which any decomposition uses $\Omega(n^2)$ tetrahedra. (In fact, the result is even stronger: any convex decomposition—a convex decomposition is a decomposition into convex pieces—uses $\Omega(n^2)$ pieces, even if one allows pieces of non-constant complexity.) Since the complexity of algorithms that need a decomposition depends on the number of tetrahedra in the decomposition, this is rather disappointing. Chazelle's polyhedron is quite special, however, and one may hope that polyhedra arising in practical applications are easier to handle. This is the topic of our paper: are there types of polyhedra that can be decomposed into fewer than a quadratic number of pieces? Erickson [9] has answered this question affirmatively for so-called *local polyhedra* by showing that any such 3-dimensional polyhedron can be decomposed into $O(n \log n)$ tetrahedra. We consider *fat polyhedra*.

Types of fatness. Before we can state our results, we first need to give the definition of fatness that we use. In the study of realistic input models [5], many definitions for fatness have been proposed. When the input is convex many of these definitions are basically equivalent. When the input is non-convex, however, this is not the case: polyhedra that are fat under one definition may not be fat under a different definition. Therefore we study two different definitions.

The first is a generalization of the (α, β)-covered objects introduced by Efrat [8] to 3-dimensional objects. A simply-connected object P in $\mathbb{R}^3$ is (α, β)-*covered* if the following condition is satisfied: for each point $p \in \partial P$ there is a simplex T with one vertex at p that is fully inside P such that T is α-fat and has diameter $\beta \cdot diam(P)$. Here a tetrahedron is called α-fat if all its solid angles are at least α, and $diam(P)$ denotes the diameter of P.

The second definition that we use was introduced by De Berg [2]. For an object o and a ball B whose center lies inside o, we define $B \sqcap o$ to be the connected component of $B \cap o$ that contains the center of B. An object o is *locally-γ-fat* if for every ball B that has its center inside o and which does not completely contain o, we have $vol(B \sqcap o) \geq \gamma \cdot vol(B)$, where $vol(\cdot)$ denotes the volume of an object. Note that if we replace $\sqcap$ with $\cap$—that is, we do not restrict the intersection to the component containing the center of B—then we get the definition of fat polyhedra

*Department of Computing Science, TU Eindhoven. P.O. Box 513, 5600 MB Eindhoven, the Netherlands. Email: {mdberg,cgray}@win.tue.nl. This research was supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 639.023.301.

proposed by Van der Stappen [13]. Also note that for convex objects the two definitions are equivalent. Hence, for convex objects we can omit the adjective "locally" from the terminology.

As observed by De Berg [2] the class of locally-γ-fat objects is strictly more general than the class of (α, β)-covered objects: any object that is (α, β)-covered for some constants α, β is also locally-γ-fat for some constant γ (depending on α, β), but the reverse is not true.

Our results. First of all we study the decomposition of (α, β)-covered polyhedra and locally-γ-fat polyhedra into tetrahedra. By modifying Chazelle's polyhedron so that it becomes (α, β)-covered, we obtain the following negative result.

- There are (α, β)-covered (and, hence, locally fat) polyhedra with n vertices such that any decomposition into convex pieces uses $\Omega(n^2)$ pieces.

Next we restrict the class of fat polyhedra further by requiring that their faces should be convex and fat,when considered as planar polygons in the plane containing them. For this class of polyhedra we obtain a positive result.

- Any locally-fat polyhedron (and, hence, any (α, β)-covered polyhedron) with n vertices whose faces are convex and fat can be decomposed into $O(n)$ tetrahedra in $O(n \log n)$ time.

Several applications that need a decomposition or covering of a polyhedron into tetrahedra would profit if the tetrahedra were fat. For instance, it would make results on ray shooting in fat convex polyhedra [1, 4] directly applicable. In the plane any fat polygon can be covered by $O(n)$ fat triangles, as shown by Van Kreveld [11] (for a slightly different definition of fatness). We show that a similar result is, unfortunately, not possible in 3-dimensional space.

- There are (α, β)-covered (and, hence, locally-fat) polyhedra with n vertices and convex fat faces such that the number of tetrahedra in any covering that only uses fat tetrahedra cannot be bounded as a function of n.

2 Decomposition into tetrahedra

In this section we discuss decomposing fat non-convex objects into tetrahedra that need not be fat themselves.

The upper bound. Let P be a locally-γ-fat polyhedron in $\mathbb{R}^3$ whose faces, when viewed as polygons in the plane containing the face, are convex and β-fat. We will prove that P can be decomposed into $O(n/\gamma\beta^3)$ tetrahedra in $O(n \log n)$ time.

In our proof, we will need the concept of density. The *density* of a set S of objects in $\mathbb{R}^3$ is defined as the smallest number λ such that the following holds: any ball $B \subset \mathbb{R}^3$ is intersected by at most λ objects $o \in S$ such that $diam(o) \geq diam(B)$.

We also need the following technical lemma. Its proof is standard and omitted from this abstract.

Lemma 1 *Let P be a convex β-fat polygon embedded in $\mathbb{R}^3$ where $diam(P) \geq 1$. Let C and C' be axis-aligned cubes centered at the same point. Let the side length of C be 1 and the side length of C' be $2\sqrt{3}/3$. If P intersects C, then $P' := P \cap C'$ is β'-fat for some $\beta' = \Omega(\beta)$.*

The following lemma shows that the faces of a locally-γ-fat polyhedron have low density if they are fat themselves.

Lemma 2 *Let F_P be the set of faces of a locally-γ-fat polyhedron P. If the faces of P are themselves β-fat and convex, then F_P has density $O(1/\gamma\beta^3)$.*

Proof. Let S be a sphere with unit radius. We wish to show that the number of faces $f \in F_P$ with $diam(f) \geq 1$ that intersect S is $O(1/\gamma\beta^3)$.

Partition S into eight equal-sized cubes by bisecting S along each dimension. Consider one of the cubes: call it C. Also construct an axis-aligned cube C' that has side length $2\sqrt{3}/3$ which has its center at the center of C. For all faces f intersecting C that have $diam(f) \geq 1$, we define $f' := f \cap C'$. By Lemma 1, we know that f' is β'-fat for some $\beta' = \Omega(\beta)$.

Since f' is a fat convex polygon with a diameter of at least $2\sqrt{3}/3 - 1$, it must contain a circle c of radius $\rho = \beta'(2\sqrt{3}/3 - 1)/8$ [13]. For any such circle c, there is a face f of C' such that the projection of c onto f is an ellipse which has a minor axis with length at least $\rho/\sqrt{2}$.

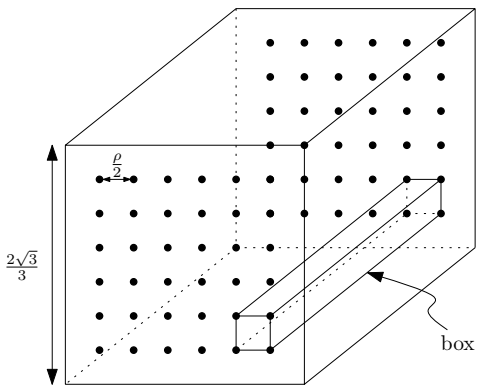


Figure 1: A box.

We make a grid on each face of C' where every grid cell has side length $\rho/2$. We call the rectangular prism

between two grid cells on opposite faces of C' a *box*—see Figure 1. Each face f' has an intersection with a box that is the entire cross-section of the box. We assign each face to such a box.

We now consider the number of faces that can be assigned to any one box b. There are two types of face in b. For example, if b has its long edges parallel to the x axis, there are the faces that have the interior of P in the positive x direction and the faces that have the interior in the negative x direction. We consider one type of face at a time. For each face f_i, we place sphere s_i with radius $\rho/2$ so that its center is on f_i and in the center of b (meaning exactly between the long faces of b). Since P is locally-γ-fat, $vol(P \cap s_i) \geq 4\gamma\pi\rho^3/6$. Since we only consider one type of face, $(P \sqcap s_i) \cap (P \sqcap s_j) = \emptyset$ for any $s_j \neq s_i$. Therefore the number of faces of one type that can cross one box is $(2\sqrt{3}/(\gamma\pi\rho))$. The number of faces that can cross one box is thus $2(2\sqrt{3}/(\gamma\pi\rho))$. The number of boxes is $(1/\rho^2)$. Hence, the number of faces that can intersect S is at most $16(2\sqrt{3}/(\gamma\pi\rho))(1/\rho^2)$. Since $\rho = O(\beta)$, this is $O(1/\gamma\beta^3)$. □

Since the set F_P of faces of the polyhedron P has density $O(1/\gamma\beta^3)$, there is a BSP for F_P of size $O(n/\gamma\beta^3)$, which can be computed in $O(n \log n)$ time [3]. The cells of a BSP are convex and contain at most one facet, so we can easily decompose all cells further into $O(n/\gamma^4)$ tetrahedra in total.

Theorem 3 *Any locally-γ-fat polyhedron with β-fat convex faces can be partitioned into $O(n/\gamma\beta^3)$ tetrahedra in $O(n \log n)$ time.*

The lower bound. Next we show that the restriction that the faces of the polyhedron are fat is necessary, because there are fat polyhedra without fat faces that need a quadratic number of tetrahedra to be covered.

The polyhedron known as *Chazelle's polyhedron* [6]—see Figure 3—is an important polyhedron used to construct lower-bound examples. We describe a slight modification of that polyhedron which makes it (α, β)-covered and which retain the properties needed for the lower bound.

The essential property of Chazelle's polyhedron is that it contains a region sandwiched between a set L of lines defined as follows:

$$L := \{y = i, z = ix - \varepsilon : i \leq n \text{ and } i \in \mathbb{N}\}$$

$$\cup \{x = i, z = iy \text{ and } i \in \mathbb{N}\},$$

where ε is a small positive real number. The region $\Sigma := \{(x, y, z) : 0 \leq x, y \leq n \text{ and } xy - \varepsilon \leq z \leq xy\}$ between these lines has volume $\Theta(\varepsilon n^2)$. Chazelle showed that for any convex object o that does not intersect any of the lines in L we have $vol(o \cap \Sigma) = O(\varepsilon)$. These two facts are enough to show that $\Omega(n^2)$ convex objects are required to cover any polyhedron that

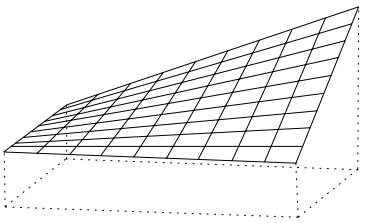


Figure 2: The lines used in the lower-bound construction.

contains Σ but whose interior does not intersect the lines in L.

With the aim of making the lines in L into edges of a fat polyhedron, we first turn the lines into line segments of length n starting either at $x = 0$ or $y = 0$. We then make each line segment into an equilateral triangular prism where the long edges have length n and the short edges have length ε'. For the lines on $z = xy$, the prism goes above the line and for the lines on $z = xy - \varepsilon$, the prism goes below the line. In this way, we ensure Σ will be contained in our polyhedron.

We then subtract the prisms that we created from a cube of appropriate size. This means that the prisms are completely contained in the cube and that the boundary of the cube contains the $x = 0$ and $y = 0$ planes. We call this cube C and the resulting polyhedron P. The polyhedron P is locally-γ-fat but not

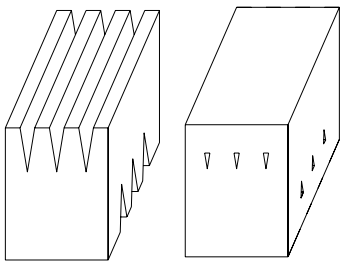


Figure 3: Chazelle's polyhedron before and after modification. The modified version remains topologically equivalent to a sphere.

(α, β)-covered. This is because the angle between a prism at $x = n$ and the boundary of C is very small (it depends on $1/n$). Therefore, a fat tetrahedron cannot fit between such a prism and the boundary of C.

We now move on to a modification of P that is (α, β)-covered. We begin with the set L of prisms. We add the set of line segments

$$B := \{y = i, z = -\varepsilon, -n \leq x \leq 0 : 1 \leq i \leq n, i \in \mathbb{N}\}$$

$$\cup \{x = i, z = 0, -n \leq y \leq 0, i \in \mathbb{N}\}$$

and transform these segments into prisms as above. The prisms in B are called "bridges". We again subtract the prisms in $L \cup B$ from an appropriately-sized

cube. In this case, that means that the cube's boundary contains the planes $x = -n$ and $y = -n$ as well as everything in $L \cup B$. We call the new cube C' and the new polyhedron P'. The bridges in B give us room to place a good tetrahedron anywhere on the boundary of P'. The formal proof of this fact is omitted.

Theorem 4 *There are constants $\alpha, \beta > 0$ such that for any large enough n there is an (α, β)-covered polyhedron with n vertices for which any decomposition into convex pieces uses $\Omega(n^2)$ pieces.*

3 Decomposition into fat tetrahedra

When we attempt to partition non-convex polyhedra into fat tetrahedra the news is uniformly bad. That is, no matter which of the realistic input models we use (of those we are studying), the number of fat tetrahedra necessary to partition the polyhedron can be made arbitrarily high. For polyhedra without fatness restrictions, there are many examples which require an arbitrary number of fat tetrahedra for partitioning. Perhaps the simplest is a rectangular box of size $1 \times (\beta/k) \times (\beta/k)$. This box requires $\Omega(k)$ β-fat tetrahedra to partition (or cover) it.

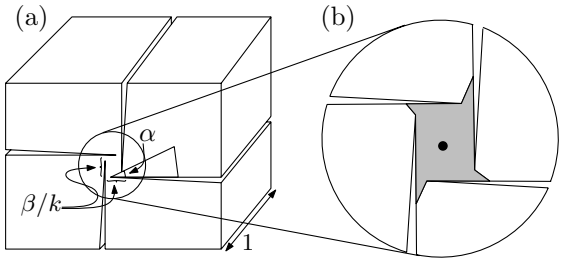


Figure 4: (a) An (α, β)-covered polyhedron with fat faces whose interior cannot be covered by a bounded number of fat tetrahedra. (b) The part of the polyhedron seen by a point in the center.

There are also (α, β)-covered polyhedra with fat faces that need an arbitrary number of fat tetrahedra to be partitioned or covered. One example is Figure 4, where the "tube" requires $\Omega(k)$ β-fat tetrahedra in any convex decomposition. The essential feature of the construction in Figure 4 is that every point along the long axis of the tube can see very little relative to the amount that points on the boundary can see.

Theorem 5 *There are (α, β)-covered polyhedra with n vertices and convex fat faces such that the number of tetrahedra in any covering that only uses fat tetrahedra cannot be bounded as a function of n.*

4 Concluding remarks

We have shown that any locally-fat polyhedron with n vertices fat convex faces can be decomposed into $O(n)$ tetrahedra, and that the restriction that the faces be fat is necessary. We also showed that one cannot obtain a decomposition using a bounded (in terms of n) number of of fat tetrahedra. In some applications—ray shooting, for instance—one does not need a decomposition of the interior of a polyhedron. Instead, a covering of only the boundary of the polyhedron suffices. It would be interesting to see if better results are possible in this case.

Acknowledgments

The second author thanks Herman Haverkort, Elena Mumford, and Bettina Speckmann for conversations regarding this topic.

References

[1] B. Aronov, M. de Berg, and C. Gray. Ray shooting and intersection searching amidst fat convex polyhedra in 3-space. *Proc. 22nd ACM Symp. Computat. Geom.*, pages 88–94, 2006.

[2] M. de Berg. Improved bounds on the union complexity of fat objects. In *Proc. 25th Conference on Foundations of Software Technology and Theoretical Computer Science*, LNCS 3821, pages 116–127, 2005.

[3] M. de Berg. Linear size binary space partitions for uncluttered scenes. *Algorithmica* 28:353–366, 2000.

[4] M. de Berg and C. Gray. Vertical ray shooting and computing depth orders for fat objects. In *Proc. 17th Annual Symposium on Discrete Algorithms*, pages 494–503, 2006.

[5] M. de Berg, A.F. van der Stappen, J. Vleugels, and M. J. Katz. Realistic input models for geometric algorithms. *Algorithmica*, 34(1):81–97, 2002.

[6] B. Chazelle. Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm. *SIAM J. Comput.* 13:488–507 (1984).

[7] B. Chazelle. Triangulating a simple polygon in linear time. *Discr. Comput. Geom.* 6:485–524 (1991).

[8] A. Efrat. The complexity of the union of (α, β)-covered objects. *SIAM J. Comput.* 34:775–787 (2005).

[9] J. Erickson. Local polyhedra and geometric graphs *Comput. Geom. Theory Appl.* 31:101–125 (2005).

[10] J.M. Keil. Polygon Decomposition. In: J.-R. Sack and J. Urrutia (eds.). *Handbook of Computational Geometry*, pages 491–518, 2000.

[11] M. van Kreveld. On fat partitioning, fat covering, and the union size of polygons. *Comput. Geom. Theory Appl.* 9:197–210 (1998).

[12] J. Rupert and R. Seidel. On the difficulty of triangulating three-dimensional nonconvex polyhedra. *Discr. Comput. Geom.* 7:227–253 (1992).

[13] A.F. van der Stappen. *Motion planning amidst fat obstacles.* Ph.D. thesis, Utrecht University, Utrecht, the Netherlands, 1994.

Schnyder Woods for Higher Genus Triangulated Surfaces

Luca Castelli Aleardi* Eric Fusy† Thomas Lewiner‡

Abstract

We study a well known characterization of planar graphs, also called Schnyder wood or Schnyder labelling, which yields a decomposition into vertex spanning trees. The goal is to extend previous algorithms and characterizations designed for planar graphs (corresponding to combinatorial surfaces with the topology of the sphere, i.e., of genus 0) to the more general case of graphs embedded on surfaces of arbitrary genus. First, we define a new traversal order of the vertices of a triangulated surface of genus g together with an orientation and coloring of the edges that extends the one proposed by Schnyder for the planar case. As a by-product we show how to characterize our edge coloration in terms of genus g maps. All the algorithms presented here have linear time complexity.

1 Introduction

Schnyder woods are a nice and deep combinatorial structure to finely capture the notion of planarity of a graph. They are named after W. Schnyder, who introduced these structures under the name of realizers and derived as main applications a new planarity criterion in terms of poset dimensions [16], as well as a very elegant and simple straight-line drawing algorithm [17]. There are several equivalent formulations of Schnyder woods, either in term of *angle labelling* (Schnyder labellings) or *edge colouring and orientation* or in terms of orientations with prescribed degrees. The most classical formulation is for the family of maximal plane graphs, i.e., plane triangulations, yielding the following striking property: the internal edges of a triangulation can be partitioned into three spanning trees rooted respectively at each of the three vertices incident to the outer face. From the combinatorial point of view the set of Schnyder woods of a fixed triangulation has an interesting lattice structure [5, 2, 11, 8, 9], and the nice characterization in term of spanning trees motivated a large number of applications in several domains as graph drawing [17, 14], graph coding and sampling [7, 3, 15, 12, 6, 1]. Previous work focused mainly on the application and extension of the combinatorial properties of Schnyder woods to 3-connected plane graphs [10, 14]. We deal with combinatorial surfaces possibly having handles, i.e., oriented surfaces of arbitrary genus $g \geq 0$. We show how to extend the local properties of Schnyder labelling in a coherent manner to triangulated surfaces.

Spanning tree decompositions in higher genus

In the area of tree decompositions of graphs there exist some works dealing with the higher genus case. We mention one attempt to generalize Schnyder woods to the case of toroidal graphs [4], based on a special planarizing procedure. In the triangular case it is possible to obtain a partition of the edges into three edge-disjoint spanning trees plus at most 3 edges. Unfortunately, the local properties of Schnyder woods are possibly not satisfied for a large number of vertices, and it is not clear how to generalize to genus $g \geq 2$.

Contributions

Our first result consists in defining new traversal orders of the vertices of a triangulation of genus g, as extension of the *canonical orderings* defined for planar graphs. We are also able to provide a generalization of the Schnyder labelling to the case of higher genus surfaces. The major novelty is in the way we show that the linear time algorithm designed for the planar case can be extended in a nontrivial way in order to design a traversal of a genus g surface. This induces a special edge colouring and orientation that is a natural generalization of the corresponding planar structure. In particular, the spanning property characterizing Schnyder woods is again verified almost everywhere in the genus g case.

Finally, we characterize our graph decomposition in terms of maps of genus g (a natural generalization of plane trees).

1.1 Schnyder woods: definitions

Schnyder woods for plane triangulations

The definition of Schnyder woods is given in terms of local conditions, and leads to a partition of the edges into 3 spanning trees.

*Department of Computer Science, ULB University, Bruxelles, luca.castelli.aleardi@ulb.ac.be

†Laboratoire d'Informatique, Ecole Polytechnique, France, fusy@lix.polytechnique.fr

‡Departement of mathematics, PUC University, Rio de Janeiro, tomlew@mat.puc-rio.br

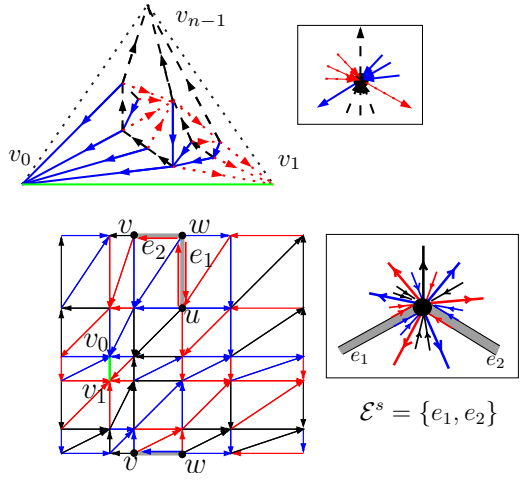


Figure 1: (above) A planar triangulation endowed with a Schnyder wood. (below) A triangulated torus with a g-Schnyder wood, together with the generalized local condition. Both these maps are rooted (green edge).

Definition 1 ([17]) *Let T be a plane triangulation with outer face (v_0, v_1, v_{n-1}), and let $\mathcal{E}$ be the set of inner edges of T. A* ***Schnyder wood*** *of $\mathcal{T}$ is an orientation and labelling, with label in $\{0, 1, 2\}$ of the edges in $\mathcal{E}$ such that the two following conditions are verified: (1) (**root face condition**) the edges incident to the vertices v_0, v_1, v_{n-1} are all ingoing and are respectively of colour 0, 1, and 2. (2) (**local condition**) For each vertex v not incident to the root face, the edges incident to v in ccw order are: one outgoing edge coloured 0, zero or more incoming edges coloured 2, one outgoing edge coloured 1, zero or more incoming edges coloured 0, one outgoing edge coloured 2, and zero or more incoming edges coloured 1, which we write concisely as*

$$(\text{Seq}(\text{In } 1), \text{Out } 0, \text{Seq}(\text{In } 2), \text{Out } 1, \text{Seq}(\text{In } 0), \text{Out } 2).$$

Generalized Schnyder woods

One main contribution is to propose a new generalized version of Schnyder woods to genus g triangulations.

Definition 2 *Let $\mathcal{S}$ be a triangulation of genus g, with n vertices; let $\mathcal{E}$ be the set of edges of $\mathcal{S}$ except those three incident to the root face (v_0, v_1, v_{n-1}). A* genus-g Schnyder wood *is a partition of $\mathcal{E}$ into a set of* normal edges *and a set $\mathcal{E}^s = \{e_1, \ldots, e_{2g}\}$ of $2g$* special edges *considered as fat, i.e., made of two parallel edges. In addition, each edge, a normal edge or one of the two edges of a special edge, is simply oriented and has a label in $\{0, 1, 2\}$, such that:*
• ***root face condition:*** *All edges incident to v_0, v_1, and v_{n-1} are ingoing of color 0, 1, and 2.*
• ***local condition for vertices not incident to special edges:*** *for every vertex $v \in \mathcal{S} \setminus \{v_0, v_1, v_{n-1}\}$ not incident to any special edge, its edges incident in ccw order are of the form:*

$$(\text{Seq}(\text{In } 1), \text{Out } 0, \text{Seq}(\text{In } 2), \text{Out } 1, \text{Seq}(\text{In } 0), \text{Out } 2).$$

• ***local condition for vertices incident to special edges:*** *A vertex v incident to $k \geq 1$ special edges has exactly one outgoing edge in colour 2. Consider the $k+1$ sectors around v delimited by the k special edges and the outgoing edge in colour 2. Then in each sector, the edges occur as follows in cw order:* $\text{Seq}(\text{In } 1), \text{Out } 0, \text{Seq}(\text{In } 2), \text{Out } 1, \text{Seq}(\text{In } 0)$.

The planar and the genus g definition do coincide in the planar case (see Figure 1 for an example). Moreover, the local condition is again true almost everywhere (except the few vertices lying in $\mathcal{E}^s$): almost all the vertices have out-degree 3, thus the g-Schnyder woods are good characterization of the local planarity of a bounded genus surface.

2 Computing Schnyder woods in higher genus

2.1 Handle operators: notations and definitions

As in the planar case, our strategy consists in `conquer`ing the whole graph incrementally, face by face, using a vertex-based operator (`conquer`) and two new operators (`split` and `merge`) designed to represent the handle attachments. Given a triangulated surface $\mathcal{S}$ having genus g and n vertices, we denote by S^{out} (S^{in}) the subgraph of $\mathcal{S}$ induced by the faces already conquered (not yet conquered, respectively). S^{out} is a face-connected map of genus g having $b \geq 1$ boundaries, each boundary being a simple cycle C_i, $i \in 1 \ldots b$. We define $\partial S^{in} := \cup_{i=1}^{b} C_i$ as the overall border between S^{in} and S^{out}.

Definition 3 *A* chordal edge *is an edge of $S^{out} \setminus \partial S^{in}$ whose two extremities are on ∂S^{in}. A boundary vertex $w \in C_i$ is* free *if w is not incident to a chordal edge e.*

The operator `conquer`, does not modify the topology of S^{in}: the *conquest* of a free vertex w consists in transferring from S^{out} to S^{in} all faces incident to w that were not yet in S^{in}. Given $\mathcal{S}$ and a collection of b cycles $\{C_i\}$ delimiting a face-connected map S^{out}, a chordal edge e for S^{out} is said to be *nonseparating* if S^{out} is not disconnected when cutting along e. A chordal edge with extremities in the same cycle C_i is a *splitting chordal edge*. Let C' and C'' be the two cycles formed by $C_i + e$. Then e is said to be *contractible* if either C' or C'' is contractible.

Definition 4 (split and merge edges) *A* split edge *for the area S^{out} is a nonseparating splitting chordal edge. A* merge edge *for the area S^{out} is a nonseparating chordal edge e having its two extremities on two distinct cycles C_i and C_j, $i \neq j$.*

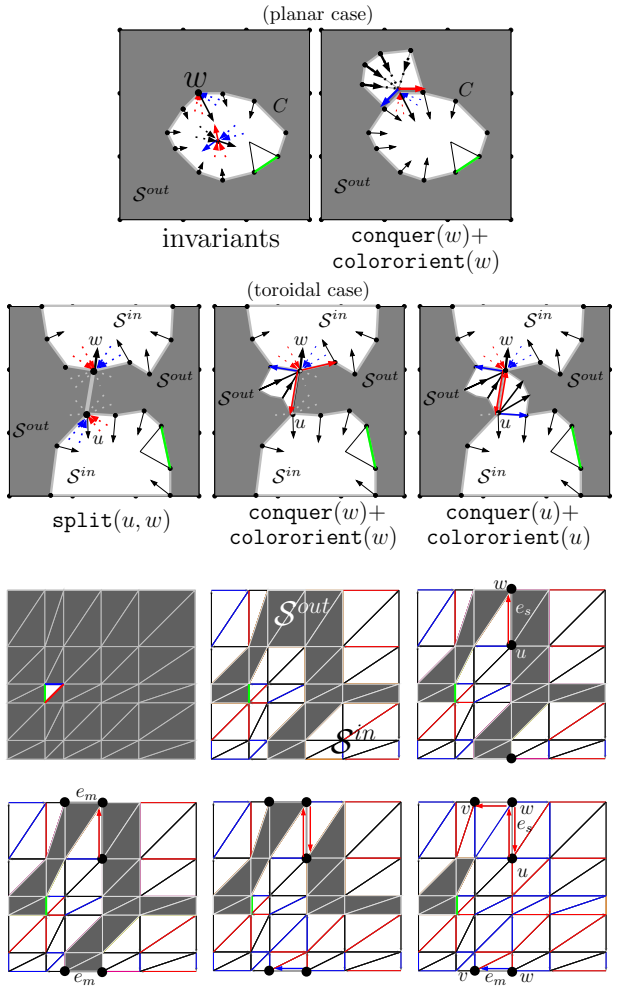


Figure 2: (above) The first pictures show the result of a `colorient` operation, both in the planar and higher genus case. (below) An execution of our traversal algorithm for a triangulated surface of genus 1. When there remain no free vertices to conquer, it is possible to find *split* and *merge* edges. After performing split and merge operations, new free vertices can be found in order to continue the traversal.

The `split` and `merge` operators defined below are (the inverse of) handle operators of type 1: they are designed to identify boundary vertices lying on distinct cycles. Intuitively, the `split` operator *splits* a boundary curve into two distinct components, thereby increasing the number of boundaries of ∂S^{in} by 1; while a *merge* operator merges two boundaries, thereby decreasing the the number of boundaries of ∂S^{in} by 1.

Given a split edge $e = (u, w)$ having its extremities on a cycle C, the operator `split`(e) produces the splitting of $C = \{v_0, v_1, \ldots, v_k\}$ (with $u = v_{i_1}$ and $w = v_{i_2}$, for some indices $i_1 < i_2 \leq k$) into two new cycles C', C''. A `merge` operation can be performed in a similar way, on a given merge edge $e = (v'_{i_1}, v''_{i_2})$ with extremities on different boundaries C' and C'', and produces a new cycle $\overline{C}$ containing two copies of v'_{i_1} and v''_{i_2} and all the vertices in C' and C''.

The edges concerned by the merge/split operations are exactly the special edges involved in the definition of the genus g Schnyder wood.

2.2 A new traversal algorithm for genus g surfaces

Once defined the `conquer` operation, we can associate to it a simple rule for colouring and orienting the edges incident to a vertex conquered. The `colorient`(v) is defined as follows: orient outward of v the two edges connecting v to its two neighbours on ∂S^{in}; assign color 0 (1) to the edge connected to the left (right, respectively) neighbour, looking toward S^{out}. Orient toward v and color 2 all edges incident to v in $S^{out} \backslash \partial S^{in}$. As in the planar case, we can now formulate the algorithm for computing a Schnyder wood as a sequence of $n-2$ `conquer` and `colorient` operations (as suggested by Brehm [5]). Here the important difference is that the `conquer` operations are interleaved with $2g$ merge/split operations so as to make the genus of the conquered area increase from 0 to g. At the beginning S^{in} is a topological disk delimited by the simple cycle $C_0 := \{(v_0, v_1, v_{n-1})\}$.

ComputeSchnyderAnyGenus($\mathcal{S}$)
($\mathcal{S}$ a triangulated surface of genus g)
`while` $\{\mathcal{C}\} \neq \{v_0, v_1\}$
  `If there is a free vertex` v `on some` $C_i \in \mathcal{C}$
    `conquer(`v`)+colorient(`v`);`
  `otherwise, if there exists a split edge` e
    `split(e); add a new cycle` C' `to` $\mathcal{C}$;
  `otherwise, find a merge edge` e
    `merge(e); remove a cycle from` $\mathcal{C}$;
`end while`

The correctness and termination of the traversal algorithm above is based on the fundamental property that either there exists a free vertex, or we can find split or merge edges.

Lemma 1 *For any triangulated surface $\mathcal{S}$ of genus g,* ComputeSchnyderAnyGenus*($\mathcal{S}$) terminates, and can be implemented to run in linear time.*

One major advantage of computing a Schnyder wood in an incremental way is that we are able to put in evidence some invariants, which remain satisfied at each step of the algorithm. This allows to provide a characterization in terms of genus g maps, which is an extension of the fundamental property of planar Schnyder woods [17], stating that T_0, T_1, and T_2 are plane trees.

Theorem 2 *Then the coloring and orientation of edges computed by* ComputeSchnyderAnyGenus *is a genus g Schnyder wood. The Schnyder wood thus*

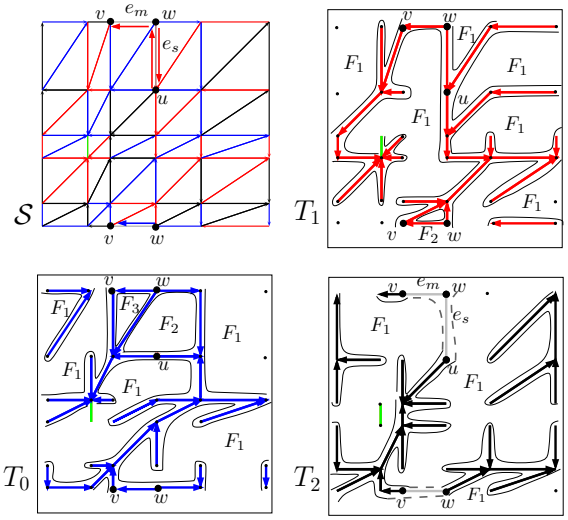


Figure 3: A triangulated torus endowed with a Schnyder wood, together with the maps T_0 (a map with 3 faces), T_1 (a map with 2 faces) and T_2 (which is a map with 1 face, when adding the two special edges).

obtained has the additional property that the graph T_2 formed by the edges of color 2 is a tree, and the embedded graph formed by T_2 and the $2g$ special edges is a one-face map; moreover the embedded graphs T_0, T_1 formed by the edges of color 0 and of color 1 are genus g maps with at most $1 + 2g$ faces.

3 Conclusion and perspectives

We have presented a general approach for extending to higher genus a fundamental combinatorial structure, Schnyder woods, which is by now a standard tool to handle planar graphs both structurally and algorithmically. We have been successful in showing that the definition and several fundamental combinatorial properties can be extended from the planar to the genus g case in a natural way. Our work leaves several interesting questions open. Let us recall that, in the planar case, the set of Schnyder woods of a fixed triangulation is a distributive lattice; in addition the Schnyder wood at the bottom of the lattice is a key ingredient in a bijective optimal encoder for plane triangulations [15]. We would like to investigate the extension of these properties in the higher genus case. More generally, we think that the local properties of genus g Schnyder woods (Definition 2) suggest the possibility of further nice applications in graph encoding and sampling (see [7, 13, 12, 15] for the planar case).

4 Acknowledgements

We are grateful to N. Bonichon, C. Gavoille and A. Labourel for enlightening discussions on Schnyder woods. First author would like to thank E. Colin de Verdière for pointing out some useful topological properties of graphs on surfaces. We are extremely grateful to G. Schaeffer for enlightening discussions on the combinatorics of maps which motivated this work.

References

[1] J. Barbay, L. Castelli-Aleardi, M. He, and J. I. Munro. Succinct representation of labeled graphs. In *ISAaC*, pages 316–328, 2007.

[2] N. Bonichon. *Aspects algorithmiques et combinatoires des réaliseurs des graphes plans maximaux*. PhD thesis, Bordeaux I, 2002.

[3] N. Bonichon, C. Gavoille, and N. Hanusse. An information-theoretic upper bound of planar graphs using triangulation. In *STACS*, pages 499–510. Springer, 2003.

[4] N. Bonichon, C. Gavoille, and A. Labourel. Edge partition of toroidal graphs into forests in linear time. In *ICGT*, volume 22, pages 421–425, 2005.

[5] E. Brehm. 3-orientations and Schnyder 3-Tree decompositions. Master's thesis, FUB, 2000.

[6] L. Castelli-Aleardi, O. Devillers, and G. Schaeffer. Optimal succinct representations of planar maps. In *SoCG*, pages 309–318, 2006.

[7] R. C.-N. Chuang, A. Garg, X. He, M.-Y. Kao, and H.-I. Lu. Compact encodings of planar graphs via canonical orderings and multiple parentheses. *ICALP*, pages 118–129, 1998.

[8] H. de Fraysseix and P. O. de Mendez. On topological aspects of orientations. *Disc. Math.*, 229:57–72, 2001.

[9] P. O. de Mendez. *Orientations bipolaires*. PhD thesis, Paris, 1994.

[10] S. Felsner. Convex drawings of planar graphs and the order dimension of 3-polytopes. *Order*, 18:19–37, 2001.

[11] S. Felsner. Lattice structures from planar graphs. *Electron. J. of Combinatorics*, 11(15):24, 2004.

[12] E. Fusy, D. Poulalhon, and G. Schaeffer. Dissections and trees, with applications to optimal mesh encoding and to random sampling. In *SoDA*, pages 690–699, 2005.

[13] X. He, M.-Y. Kao, and H.-I. Lu. Linear-time succint encodings of planar graphs via canonical orderings. *Journal on Discrete Mathematics*, 12:317–325, 1999.

[14] G. Kant. Drawing planar graphs using the canonical ordering. *Algorithmica*, 16(1):4–32, 1996.

[15] D. Poulalhon and G. Schaeffer. Optimal coding and sampling of triangulations. *Algorithmica*, 46:505–527, 2006.

[16] W. Schnyder. Planar graphs and poset dimension. *Order*, pages 323–343, 1989.

[17] W. Schnyder. Embedding planar graphs on the grid. In *SoDA*, pages 138–148, 1990.

Seed Polytopes for Incremental Approximation *

Oswin Aichholzer† Franz Aurenhammer‡ Thomas Hackl† Bernhard Kornberger†
Simon Plantinga§ Günter Rote¶ Astrid Sturm¶ Gert Vegter§

Abstract

Approximating a given three-dimensional object in order to simplify its handling is a classical topic in computational geometry and related fields. A typical approach is based on incremental approximation algorithms, which start with a small and topologically correct polytope representation (the seed polytope) of a given sample point cloud or input mesh. In addition, a correspondence between the faces of the polytope and the respective regions of the object boundary is needed to guarantee correctness.

We construct such a polytope by first computing a simplified though still homotopy equivalent medial axis transform of the input object. Then, we inflate this medial axis to a polytope of small size. Since our approximation maintains topology, the simplified medial axis transform is also useful for skin surfaces and envelope surfaces.

1 Introduction

Object simplification and surface reconstruction are fundamental tasks in several areas of computer science, like geometric modeling, computer graphics, and computational geometry. We refrain from a general discussion here and refer the reader e.g. to [3, 8, 9, 11] and references therein.

In this note we deal with the problem of computing a simple but topologically correct polytope for a given input object, which is typically presented as a point cloud or surface mesh. As this polytope will serve as a starting point for incremental approximation algorithms, we additionally provide a correspondence between the faces of the polytope and the regions of the object surface. Possible incremental algorithms we have in mind use e.g. elliptical and hyperbolic patches or are based on interpolating subdivision surfaces.

In a previous related approach, point clouds in *convex* position are approximated by spherical patches, using an incremental algorithm [7]. Starting with a very simple structure (a tetrahedron), the convex polytope is incrementally refined until the associated surface built from spherical patches approximates the convex point cloud within a given tolerance bound. The approximating surface consists of a 'bulgy' polytope, where the triangular faces of the polytope are replaced by spherical patches.

The same approach works for other classes of approximating surfaces based on polytopes, such as interpolating subdivision surfaces. For a correct approximation, the underlying polytope needs to have the same topology as the input object. Furthermore, one has to be able to find which part of the object is approximated by a given part of the polytope, as this is the area where we have to test for epsilon-closeness.

1.1 A new approach

Our construction of a small (in general, nonconvex) initial polytope for a given, sufficiently dense sample point cloud is based on a certified simplification of the medial axis transform (MAT). The goal is to represent the object with as few elements as possible. To this end, we use a modification of our previous work [1, 2] where the input object is approximated by a set of balls. This set is then pruned based on an approximation of the minimal set covering problem, thereby carefully choosing the parameters of the original algorithm in order to preserve topology, see Section 2. With slight modifications, this approach can also be used for simplification of skin surfaces [9] and envelope surfaces [11]. The exact medial axis of the pruned set of balls is then computed [5].

In a second step we 'inflate' the simplified medial axis (which, as being defined by a union of balls, is a piecewise-linear object) by replacing it with a combinatorial 2-manifold and moving its vertices back to the input surface, see Section 3.

From experimental results for the medial axis simplification we expect that our approach leads to incremental approximations with significantly fewer patches compared to results achievable when starting directly with the original input set.

*Partially supported by the FWF Joint Research Program 'Industrial Geometry' S9205-N12, and by the IST Programme of the EU as a Shared-cost RTD (FET Open) Project under Contract No IST-006413 (ACS – Algorithms for Complex Shapes)

†Institute for Software Technology, Graz University of Technology, Austria, {oaich,thackl,bkorn}@ist.tugraz.at

‡Institute for Theoretical Computer Science, Graz University of Technology, Austria, auren@igi.tugraz.at

§University of Groningen, Department of Mathematics and Computing Science {simon,gert}@cs.rug.nl

¶Institut für Informatik, Freie Universität Berlin {rote,sturm}@inf.fu-berlin.de

2 Medial axis extraction

Let O be a smooth and boundary-connected object in 3D. We allow O to have tunnels, but 'holes' (empty regions within the object, e.g. bubbles in a Swiss cheese, without connection to the exterior) are excluded. The medial axis of O is the set of centers of all maximal inscribed balls. The local feature size $f(x)$ of a point x on the boundary ∂O of O is the minimum distance from x to any point on the medial axis of O. A finite point set $S \subset \partial O$ is an r-sample [3] of ∂O if every point $x \in \partial O$ has at least one point of S within distance $r \cdot f(x)$. We are interested in a simplified version of the medial axis of O which, nevertheless, retains two important properties: inclusion in the object, and homotopy equivalence. For an example see Figure 1 which shows the discrete MAT of a cow model and its simplified version.

2.1 Ball generation

In a first step we follow well known paths [4] in that we compute the Voronoi diagram of a given sample S of ∂O and extract all inner polar balls. Each point $s \in S$ defines an inner polar ball $b_{c,\rho}$ whose center c is a vertex of the Voronoi cell for s farthest away from s and inside O, and whose radius is $\rho = \delta(c, s)$ (the distance between c and s). Let B be the set of all inner polar balls. As has been shown in [4], the medial axis of the union, $U(B)$, of the balls in B is homotopy equivalent to O as long as S satisfies the sampling condition, that is, S constitutes an r-sample of ∂O for sufficiently small r.

In certain applications we are given not only an unorganized point set S but a triangular mesh representing ∂O and having S as its vertices. This form of input will allow the ball generation algorithm in [1] to work well even if S does not satisfy any sampling condition. Guarantees on the topology are then, of course, lost.

2.2 Pruning

The sampling density of S may cause the set, B, of balls to be quite large, so the medial axis of $U(B)$ is likely to contain many detailed and unwanted features. Therefore we do not directly compute the medial axis of $U(B)$ but perform a pruning of B first. Several pruning criteria based on proximity and angles have been proposed, e.g., in [10, 12]. In the work [1] a method is described that is capable of discarding balls belonging to unstable parts of the medial axis without any geometric criteria. This method can be adapted to keep control over the topology of the medial axis [2]. Loosely speaking, we enlarge all the balls in B and treat them together with S as an instance of the well-known set covering problem, as is briefly described below.

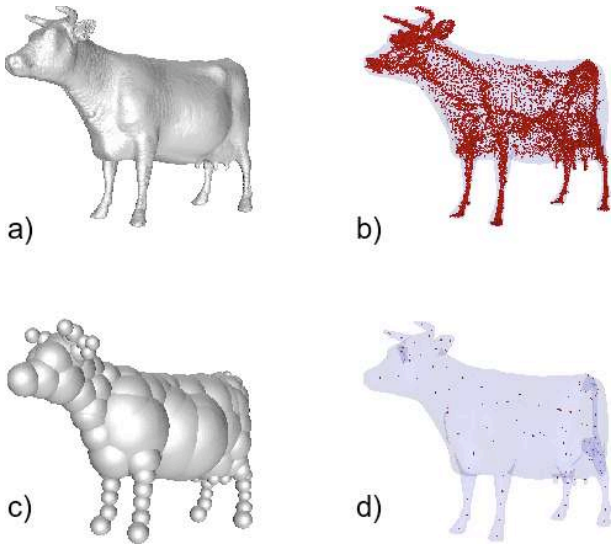


Figure 1: a,b) MAT and its 20108 medial ball centers c,d) Pruned MAT with 116 ball centers

2.2.1 Ball enlargement

By construction, each ball $b \in B$ contains 4 points of S on its boundary and has no points of S in its interior. From B we now generate a set, B', of co-centric but enlarged balls, each typically covering tens or even hundreds of points of S. Thereby, a requirement important for later purposes is that $U(B')$ and $U(B)$ are topologically equivalent. We use the power diagram $PD(B)$ of B (see Figure 2) to control the proper enlargement of the ball radii. For each ball $b \in B$, its power cell $C(b)$ contains exactly those parts of b's boundary which contribute to $\partial U(B)$, see e.g. [6]. So, if we choose maximal radii such that (1) $PD(B') = PD(B)$ holds, and (2) each $b' \in B'$ intersects the same facets, edges, and vertices of $C(b)$ as does its original b, then the topology of the union of balls does not change. Such radii exist and can be found in time linear in the size of $PD(B)$, by exploiting the well-known polytope lifting of $PD(B)$ in 4D.

2.2.2 Set covering

Now we want to keep an (ideally) minimal subset of the set B' of enlarged balls, such that all points of S are still covered by at least one ball in this subset. This is an instance of the NP-hard set covering problem. In [1] we use a combination of exact and heuristic methods in order to get an almost minimal subset $B_o \subset B'$.

Concerning the topology of the union $U(B_o)$, the set covering step only removes balls from the set B' and thus it will never close tunnels that are present in $U(B')$. (Holes do not exist in $U(B')$ by construction.) However, this step might create holes and tun-

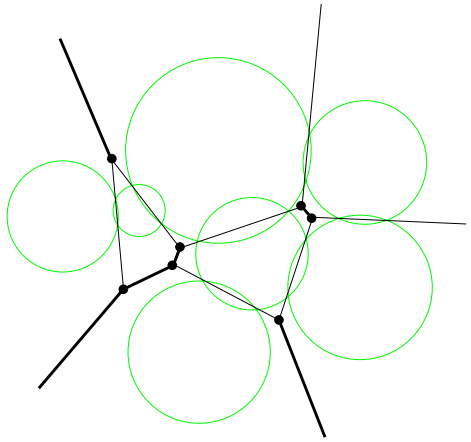


Figure 2: Power diagram and its external graph

nels in $U(B_o)$, and may even make it break apart. We therefore apply a postprocessing where such events are detected and repaired. Again, we make use of a power diagram, $PD(B_o)$, in this case. Note that disconnectedness of $U(B_o)$ can be checked from the dual graph of $PD(B_o)$ right away.

Define the *external power graph*, $G(B_o)$, of B_o as the set of all edges and vertices of $PD(B_o)$ that are completely avoided by $U(B_o)$. (All objects are considered to be topologically closed.) Consult Figure 2, where $G(B_o)$ is drawn with bold lines. Each hole in $U(B_o)$ can be detected by recognizing that $G(B_o)$ contains a respective connected component that is bounded.

To deal with tunnels, two strategies can be applied. One is to avoid tunnels altogether, by a modification of the pruning strategy for the set B' of enlarged balls: Exploiting that the input point cloud S is an r-sample, we (conceptually) shrink each ball $b \in B'$ by $r \cdot f$, where $f = \max_{x \in S \cap b} f(x)$, and execute the set covering as if such balls were present. This may lead to a (moderate) increase of the size of the pruned set B_o. On the other hand, if a mesh on S is present, then we can check for tunnels with its aid, because for each tunnel of $U(B_o)$ there exists at least one edge in $G(B_o)$ that intersects some triangle of the mesh. Starting from each such triangle, we trace $G(B_o)$ inside the mesh until we run out of edges or intersect the mesh again, in which case a tunnel has been detected. Note that most mesh triangles can be excluded from consideration; e.g. all those being covered by a single ball.

3 Construction of the polytope

The main use of a simplified polytope is to supply a good starting configuration for incremental surface approximation algorithms. To this end, we base the construction of this polytope on the pruned, piecewise-linear medial axis, $M(B_o)$, obtained in the

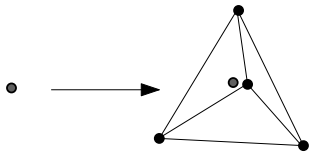


Figure 3: Constructing a pyramid from a vertex

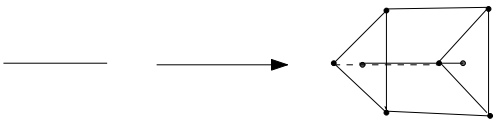


Figure 4: Constructing a tube from a segment

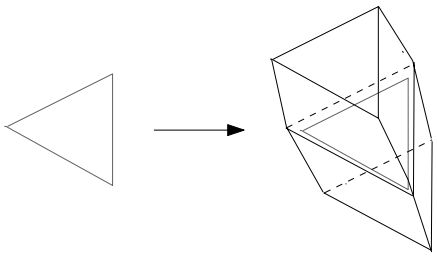


Figure 5: Constructing a polytope from a facet

previous section. The basic idea is to blow up $M(B_o)$ to a polytope, P_M, which uses only vertices of the original point cloud S. A main advantage of our construction is that the power cells of B_o give a decomposition of the space into cells, which define for each facet of P_M the neighborhood in which points from S have to be checked for epsilon-closeness to the approximating surface. This is especially important for point clouds not in convex position, since points can be very close to a surface patch then, but lie on the 'opposite' side of the medial axis, so they have to be handled by a different part of the polytope.

Note that a main condition for the constructed polytope is that $M(B_o)$ lies inside it. Therefore no polytope facet intersects the medial axis and no bound on the distance of the original vertices to the facets of the new polytope exists.

To start with, we wrap $M(B_o)$ with a combinatorial 2-manifold mesh. This wrapping will result in a mesh that is topologically equivalent to the boundary of the original input object. As $M(B_o)$ is a piece-wise linear structure, it consist of vertices, segments, and facets with boundary-segments. For each of these features, we construct a polytope feature:

- For a *vertex* we construct a pyramid (Figure 3).
- For a *segment* we construct a tube (Figure 4).
- We double a *facet* and connect it using the fea-

tures of the boundary segments (Figure 5).

We obtain a combinatorial 2-manifold mesh with its vertices still coinciding with the vertices of $M(B_o)$. The next step is to select a point of S for every vertex of the inflated medial axis P_M to be constructed. We build a cone of size γ which depends on the local feature size of the r-sampling S. Each vertex of P_M is an apex of a cone pointing in the direction of the normal in this vertex. This cone gives the direction in which we move the vertex of the wrapped medial axis towards the object boundary. The cones are chosen in such a way that they do not intersect $M(B_o)$. Moreover, the way we define the cone size (namely, depending on the local feature size) assures that each cone includes at least one point of S. If more than one point of S is included, we choose an arbitrary one. This results in a polytope containing $M(B_o)$ and with vertices chosen from the set S. The facets of this polytope are similar to the *supertriangles* as defined in [7], which can be used as starting facets for any incremental approximation algorithm.

Figure 6 summarizes our polytope construction for a (two-dimensional) point sample.

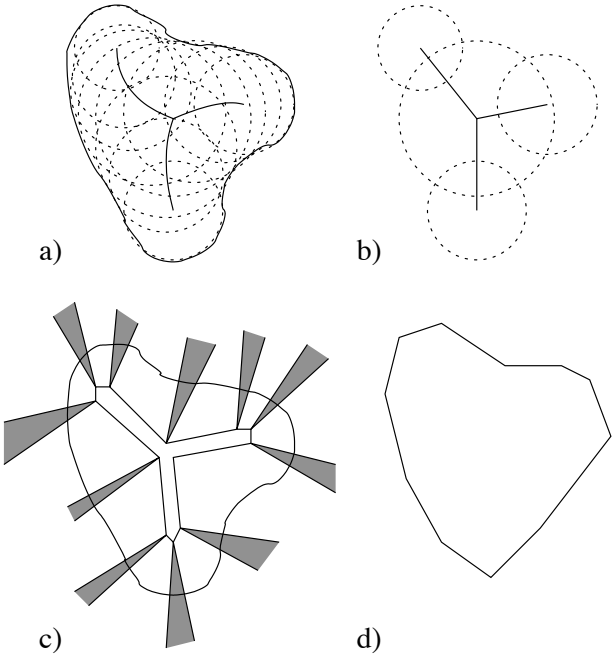


Figure 6: a) Medial axis transform, b) Pruned MAT, c) wrapped MA with cones, d) inflated MA

4 Future work

For convex objects, the spherical patch algorithm described in [7] can be used together with our setting. For the more general case of non-convex inputs we plan to extend [7] to use a combination of e.g. elliptical and hyperbolic patches, based on the presented framework. Adapting the growing strategy, the first part of our algorithm can also be useful for the skin surface algorithm as well as envelope surfaces.

References

[1] O. Aichholzer, F. Aurenhammer, T. Hackl, B. Kornberger, M. Peternell, H. Pottmann. *Approximating boundary-triangulated objects with balls.* Proc. 23rd European Workshop on Computational Geometry, EuroCG'07, 2007, 130-133.

[2] O. Aichholzer, F. Aurenhammer, T. Hackl, B. Kornberger. *Scaleable piecewise linear approximations of 3D medial axes.* Manuscript.

[3] N. Amenta, M. Bern. *Surface reconstruction by Voronoi filtering.* Discrete & Computational Geometry 22 (1999), 481-504.

[4] N. Amenta, R. Kolluri *Accurate and efficient unions of balls.* Proc. 16th Ann. ACM Symp. Computational Geometry, 2000, 119-128.

[5] D. Attali, A. Montanvert. *Computing and simplifying 2D and 3D continuous skeletons.* Computer Vision and Image Understanding 67 (1997), 261-273.

[6] F. Aurenhammer. *Improved algorithms for discs and balls using power diagrams.* Journal of Algorithms 9 (1988), 151-161.

[7] K. Buchin, S. Plantinga, G. Rote, A. Sturm, G. Vegter. *Convex approximation by spherical patches.* Proc. 23rd European Workshop on Computational Geometry, EuroCG'07, 2007, 26-29.

[8] T.K. Dey. *Curve and surface reconstruction.* In: Handbook of Discrete and Computational Geometry, (J.E. Goodman and J.O'Rourke, eds.), CRC Press, Vol. 2, 2004.

[9] H. Edelsbrunner. *Deformable smooth surface design.* Discrete and Computational Geometry 21 (1999), 87-115.

[10] M. Foskey, M.C. Lin, D. Manocha. *Efficient computation of a simplified medial axis.* Proc. 8th ACM Symp. Solid Modeling and Applications, 2003, 96-107.

[11] N. Kruithof, G. Vegter. *Envelope surfaces* Proc. 22nd Ann. ACM Symp. Computational Geometry, 2006, 411-420.

[12] M. Samozino, M. Alexa, P. Alliez, M. Yvinec. *Reconstruction with Voronoi centered radial basis functions.* Proc. 4th Eurographics Symp. on Geometry Processing, 2006, 51-60.

On the Reliability of Practical Point-in-Polygon Strategies*

Stefan Schirra†

Abstract

We experimentally study the reliability of geometric software for point location in simple polygons. The code we tested works very well for random query points, but it often fails for degenerate and also nearly degenerate queries. We also suggest a reliable alternative approach.

1 Introduction

Assume you would like to test points for inclusion in a simple closed polygon. Most likely, you will end up using one of the so-called practical point-in-polygon strategies instead of implementing one of the more sophisticated theoretically optimal point location data structures developed in computational geometry. Code for such practical point-in-polygon strategies is available on the WWW. This software is based on floating-point arithmetic, is very efficient, and works well for query points chosen uniformly at random inside the bounding box of the polygon. Or you might decide to use components from CGAL [2], LEDA [9] or some other software library providing code for point-in-polygon testing or more general point location queries.

As we will see in Section 3, most of the existent code produces wrong results for query points near or on the polygon edges, see also Fig. 1 where queries answered correctly are marked by a grey box ▪, false positives by a red disk ●, and false negatives by a green disk ●. If you know that the coordinates of query points and polygon vertices are inaccurate anyway, you might be willing to accept this. Unfortunately, sometimes there are errors not only for such problem-specific degenerate queries, but also for algorithm-specific degeneracies, cf. Fig. 4 in more or less rare cases. Are you still willing to accept this? What if your data is not subject to uncertainty at all? This is the case that we are most interested in. In this paper, we consider simple closed polygons and the corresponding binary point-inclusion predicate. This is the most important case and it can also be used for point location in polygons with holes. Furthermore, point-in-polygon testing is a subtask in landmarks algorithms for point location in arrangements of straight lines [7].

*Partially supported by DFG grant SCHI 858/1-1

†Otto von Guericke University, Department of Computer Science, Magdeburg, Germany, stschirr at ovgu.de

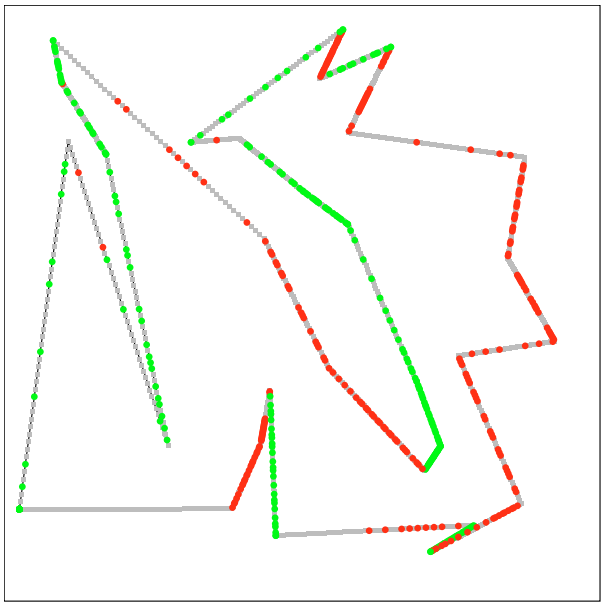


Figure 1: Results for query points near or on the edges of a random polygon with 30 edges.

After a very brief look at related work in the next section, we will report on experimental studies regarding the reliability of practical point-in-polygon testing software. The studies include code from [6], code available on the WWW, and code provided by computational geometry software libraries. Finally, we briefly discuss how to achieve full reliability without paying too much for this benefit in Section 5 .

2 Related work

Testing a query point for inclusion in a polygon is a fundamental problem in computational geometry with many applications, e.g. in computer graphics and geographic information systems, and thus has been the subject of many research papers in computer science and related application disciplines. For an overview we refer to Snoeyink's survey paper [12].

Maybe the most common algorithm for point-in-polygon testing without preprocessing is the crossing number algorithm. Interestingly, already the first description of the algorithm by Shimrat [11] contained a flaw fixed later by Hacker [5]. It is well known that handling degenerate cases in a crossing number algorithm is not obvious. Forrest [3] nicely illustrates the problems involved.

3 Non-reliability of existent code

Experimental studies on point-in-polygon testing usually focus on efficiency. In contrast, we are most interested in correctness and reliability. We concentrate on practical point-in-polygon algorithms with no or little preprocessing without sophisticated data structures. Our selection of existent code includes the fastest algorithms from the beautiful graphic gems collection of Haines [6], namely *crossings*, a "macmartinized" crossing number algorithm, see also [1], the triangle-fan algorithms *halfplane* (with sorting), *barycentric*, and *spackman*, and finally *grid*, the name says it all. Barycentric and spackman compute barycentric coordinates in addition to point location. Grid uses a 20×20 grid. Furthermore, we consider Franklin's PNPOLY code [4], which is another crossing number based algorithm, and point location code for polygons from CGAL and LEDA, where we use the latter two both with an exact and an inexact geometry kernel.

We first challenge the code with problem-dependent (near) degeneracies. We use CGAL's point generator for generating points "on" a line segment. Since we use double precision coordinates, usually not all points are exactly on the line segment, but only very close to it. Fig. 2 shows results for a real-world polygon. Besides the library codes with exact kernels all selected software produces false results. However, even with an inexact kernel based on double precision floating-point numbers, the CGAL code produces only very few false positives. Interestingly, Shimrat [11] already clearly states that his crossing number algorithm does not apply to query points on the boundary of the polygon. Haines [6] writes "*when dealing with floating-point operations on these polygons we do not care if a test point exactly on an edge is classified as being inside or outside, since these cases are extremely rare.* However, our experiments show that we get false results not only for points *exactly* on the boundary. Second, for polygons with axis-parallel edges like the H-shaped polygon in Fig. 4, points exactly on the edges are not unlikely.

Next we turn to algorithm-dependent degeneracies. We create points on the vertical and horizontal lines through the polygon vertices. These are potential degenerate cases for the crossing number algorithms. Fig. 3 shows the result for crossings and PNPOLY. Because of a conceptual perturbation, namely considering vertices on the ray as being infinitesimally above the ray, both work very well for the query points. Unfortunately, both do not produce consistent results for the vertices, in contrast to the CGAL code with an inexact kernel.

Fig. 4 shows the result for a H-shaped polygon for query points which cause algorithm-dependent degeneracies for the triangle-fan algorithms. Query points are generated "on" the non-polygon edges bounding

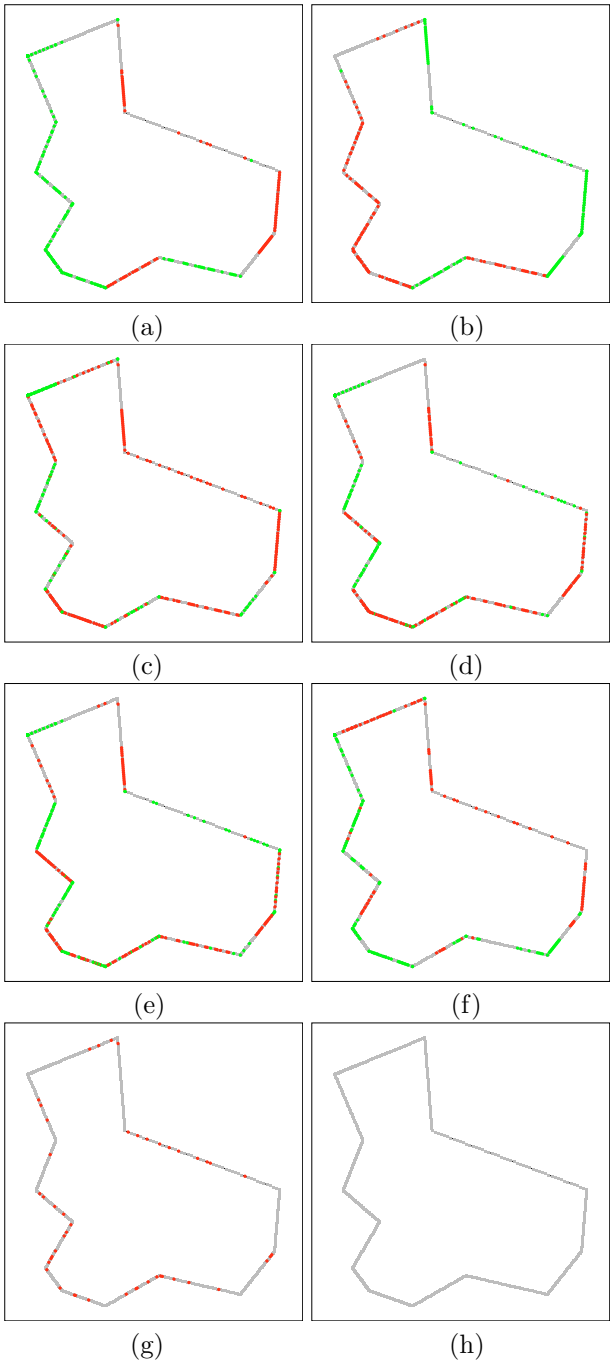
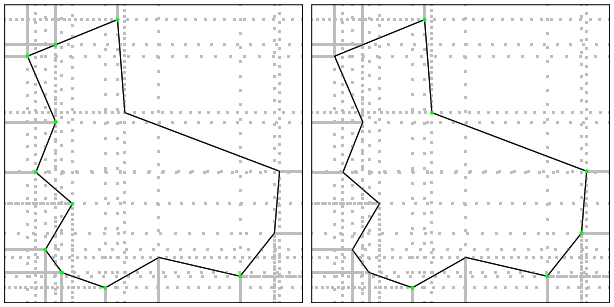


Figure 2: Results for query points on segments on a real-world polygon (simplified boundary of the village Saarwellingen in Germany): (a) crossings (b) Franklin's PNPOLY (c) halfplane (d) barycentric (e) spackman (f) grid (g) cgal with inexact kernel (h) cgal and leda with exact kernel.

the triangles considered by these algorithms. As we have suspected, the triangle-fan algorithms err for points near these edges. We have both false positive as well as false negative results, see (c), (d), and (e). Crossings (a) has false negatives on the axis-parallel edges, whereas the second crossing number algorithm



(a) (b)

Figure 3: Results for query points on verticals and horizontals through the vertices of the polygon from Fig. 2: (a) crossings (b) Franklin's PNPOLY.

has false negatives at some vertices only. Haines [6] admits that his code "*does not fully address this problem*". Again, the problems occur not only for points exactly on the triangle edges. Surprisingly, even the grid method has false negatives on axis-parallel edges as well.

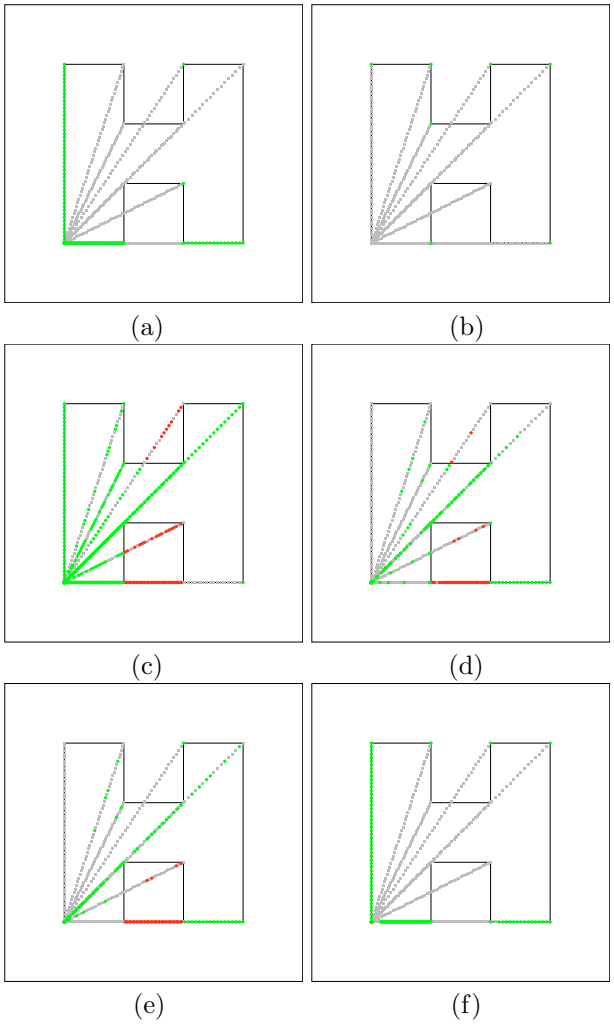
4 Reliable implementation

The straightforward approach to implement geometric algorithms like those above reliably is to use exact rational arithmetic instead of inherently imprecise floating-point arithmetic. Unfortunately, this slows down the code significantly. As suggested by the exact geometric computation paradigm [14] a better approach is to combine exact rational arithmetic with floating-point filters, e.g. interval arithmetic, in order to save most of the efficiency of floating-point arithmetic for non-degenerate cases. This approach is implemented in the exact geometry kernels of CGAL [2] and LEDA [9]. The use of adaptive predicates à la Shewchuck [10] is highly recommended.

Interestingly, exact rational arithmetic does not suffice to let crossings always produce correct results, because some degeneracies are still not handled correctly. Due to the conceptual perturbation of vertices, for some query points coincident with vertices incorrect results are still produced.

5 A reliable and efficient alternative

In terms of efficiency, algorithms with low arithmetic demand are better suited for exact geometric computation, because low demand leads to both more effective filters and less expensive rational arithmetic. With an IEEE 754 compliant floating-point arithmetic, a comparison of floating-point numbers is always exact. Thus, in terms of the cost of exact geometric computation, it pays off to replace calculations by comparisons whenever possible.



(a) (b) (c) (d) (e) (f)

Figure 4: Results for query points on segments connecting the first vertex to the remaining ones for a H-shaped polygon: (a) crossings (b) Franklin's PNPOLY (c) halfplane (d) barycentric (e) spackman (f) grid.

Let us illustrate this for the crossing number algorithm where we have to test whether a horizontal leftward ray r starting at $q = (q_x, q_y)$ intersects a segment s. This is often implemented by computing the intersection point p of the supporting line of r and the supporting line of s and then testing whether p lies on both r and s. MacMartin et al. [8] observe that s cannot intersect r if the y-coordinates of both endpoints of s are smaller or larger than q_y. Thus, we can save some calculations by additional comparisons in fortunate cases. Note that we can use comparison of x-coordinates to save further calculations as well, assuming that we did the comparison of y-coordinates already. Then, if the x-coordinates of both endpoints of s are smaller than q_x, there is no intersection, and if both are larger, there is one. If these comparisons do not suffice to decide the test, we use an exact orientation test to check whether q is to the left of s.

Next we briefly describe an alternative reliable implementation of the crossing number algorithm. We suggest to add some preprocessing to compensate for more expensive arithmetic. We use an interval skip list (or interval tree) to store the y-ranges of all non-horizontal polygon edges. In order to handle degeneracies correctly, we store half-open intervals: Only the y-coordinate of the first endpoint is included, the y-coordinate of the second endpoint is not. Here we assume that polygon edges are consistently oriented along the polygon boundary. We use another interval skip list to store the y-ranges of all vertices and all horizontal edges. Since these intervals are point intervals, we could use a multiset dictionary data structure as well. The CGAL library provides a flexible and adaptable implementation of interval skip lists which we use in our implementation. Note that all operations on the interval skip lists are exact, because we only need comparisons of floats (besides arithmetic on small integers).

To answer a query for $q = (q_x, q_y)$, we use the second interval skip list (or alternatively the dictionary data structure) to check exactly whether q lies on a horizontal ray or coincides with a polygon vertex: For all intervals containing q_y we check whether the corresponding vertex or horizontal edge contains q. If not, we use the first skip list to get candidate edges for intersection with the leftward horizontal ray starting at q and use the comparison-based strategy described above for testing for intersection. Thanks to the half-openness of the intervals, we count intersections at vertices only once.

In pathological cases we still have to consider a linear number of edges and vertices. In practice, however, we only get a few, leading to good performance for random and real-world polygons. Thanks to the preprocessing that creates the interval skip lists, we get an efficient query algorithm, where the savings due to the preprocessing compensate for the additional cost caused by applying the exact geometric computation paradigm.

6 Future work

Of course, our selection of algorithms is somewhat random. It remains to include further algorithms into this case study, especially the approach by Walker and Snoeyink, which is based on CSG-representations of polygons [13]. Furthermore, another case study will compare the efficiency of the exact counterparts of the floating-point-based practical point-in-polygon strategies we considered here and compare it to alternative approaches. Grid-based methods seem to be good candidates for achieving reliability without paying too much in terms of efficiency as well.

References

[1] T. Akenine-Möller and E. Haines. *Real-Time Rendering (2nd Ed.).* AK Peters, Ltd., 2002.

[2] CGAL, Computational Geometry Algorithms Library. `http://www.cgal.org`.

[3] A. R. Forrest. *Computational geometry in practice.* In R. A. Earnshaw, editor, Fundamental Algorithms for Computer Graphics, volume F17 of *NATO ASI*, pages 707–724. Springer-Verlag, 1985.

[4] W. R. Franklin. PNPOLY–point inclusion in polygon test. `http://www.ecse.rpi.edu/Homepages/wrf/Research/Short_Notes/pnpoly.html`.

[5] R. Hacker. Certification of algorithm 112: position of point relative to polygon. *Commun. ACM*, 5:606, 1962.

[6] E. Haines. *Point in polygon strategies.* In P. Heckbert, editor, Graphics Gems IV, pages 24–46. Academic Press, Boston, MA, 1994. `http://tog.acm.org/editors/erich/ptinpoly/`.

[7] I. Haran and D. Halperin. *An experimental study of point location in general planar arrangements.* In Proc. of ALENEX 06, pages 16–25. 2006.

[8] S. MacMartin et al. *Fastest point in polygon test.* Ray Tracing News, 5(3), 1992.

[9] K. Mehlhorn and S. Näher. *LEDA: A Platform for Combinatorial and Geometric Computing.* Cambridge University Press, Cambridge, UK, 2000.

[10] J. R. Shewchuk. *Adaptive precision floating-point arithmetic and fast robust geometric predicates.* Discrete & Computational Geometry, 18(3):305–368, 1997.

[11] M. Shimrat. Algorithm 112: position of point relative to polygon. *Commun. ACM*, 5:434, 1962.

[12] J. Snoeyink. *Point location.* In J. E. Goodman and J. O'Rourke, editors, Handbook of Discrete and Computational Geometry (2nd Ed.), chapter 34, pages 767–786. CRC Press LLC, Boca Raton, FL, 2004.

[13] R. Walker and J. Snoeyink. *Practical point-in-polygon tests using CSG representations of polygons.* In Proc. of ALENEX 99, pages 114–123. 1999.

[14] C.-K. Yap. *Towards exact geometric computation.* Comput. Geom.–Theory and Appl., 7:3–23, 1997.

Minimizing the Symmetric Difference Distance in Conic Spline Approximation

Sunayana Ghosh* Gert Vegter†

Abstract

We show that the complexity (number of elements) of an optimal parabolic or conic spline approximating a smooth curve with non vanishing curvature to within symmetric difference distance ε is $c_1\varepsilon^{-1/4} + O(1)$, if the spline consists of parabolic arcs, and $c_2\varepsilon^{-1/5} + O(1)$, if it is composed of general conic arcs of varying type. The constants c_1 and c_2 are expressed in the affine curvature of the curve. We define an *equisymmetric* bitangent conic arc to be the (unique) conic that is tangent to a curve at its endpoints, such that the areas of the two moons formed by this conic and the given curve are equal, and show that its complexity is asymptotically equal to the complexity of an optimal conic spline. We show that the symmetric difference distance between a curve and an equisymmetric conic arc tangent at its endpoints is increasing with affine arc length, provided the affine curvature along the arc is monotone. This property yields a simple bisection algorithm for computing an optimal parabolic or equisymmetric conic spline.

1 Introduction

Complexity of conic approximants. In Ghosh, Petitjean and Vegter [2] we determined the complexity, i.e., the number of elements, of parabolic and conic splines approximating a smooth planar curve to within a given Hausdorff distance. In this paper we extend this work by focusing on the *symmetric difference distance*. The symmetric difference distance of two curves that are not closed, but have common endpoints, is the total area of the regions enclosed by the two curves. See Figure 1. We show that the complexity of an optimal *parabolic spline* approximating a smooth curve to within symmetric difference distance ε, is of the form $c_1\varepsilon^{-1/4} + O(1)$. Ludwig [3] considers optimal *parabolic spline* approximation of strictly convex curves having monotone affine curvature with respect to the symmetric difference metric. Our method for computing the asymptotic error bound of an optimal parabolic spline is different from those of [3], and allows us to determine the optimal asymptotic error bound in case of general conic splines as well. Obviously, our result for parabolic splines matches those of Ludwig [3]. We also show that the complexity of approximation of a smooth curve with an optimal *conic spline* or *equisymmetric conic spline* to within symmetric difference distance ε, is of the form $c_2\varepsilon^{-1/5} + O(1)$. Here both c_1 and c_2 are expressed in terms of the affine curvature. Furthermore, for deriving the asymptotic error bounds, we use the relation between affine curvatures of the given curve and its bitangent offset curve as proved in [2, Lemma 4.1].

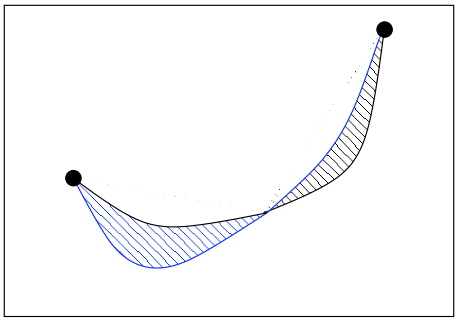


Figure 1: The symmetric difference of the two curves is the total area of the (shaded regions) two moons.

Algorithmic issues. For curves with monotone affine curvature, called *affine spirals*, we *conjecture* that there is a unique bitangent conic which minimizes the symmetric difference distance to a smooth affine spiral. However, there is another conic spline achieving the same asymptotic bound on the symmetric difference metric. More precisely, we introduce the *equisymmetric bitangent conic* of an affine spiral, which is *uniquely* determined by the fact that the two moons it forms with the affine spiral have equal area. An *equisymmetric conic spline* is a tangent continuous conic spline all elements of which are equisymmetric bitangent conics of the affine spiral. The equisymmetric conic spline has the property that all moons formed by this spline and the affine spiral have equal area, and we denote by C_{es} the spline that minimizes the symmetric difference distance to the spiral among all equisymmetric conic splines. Moreover, the complexity of this equisymmetric conic spline as a function of the symmetric difference distance to the affine spiral

*Institute of Mathematics and Computing Science, University of Groningen, S.Ghosh@cs.rug.nl

†Institute of Mathematics and Computing Science, University of Groningen, G.Vegter@cs.rug.nl

is asymptotically equal to the complexity of the optimal conic spline with respect to this error metric. Therefore, we call the computation of the optimal equisymmetric conic spline a *near-optimal approximation scheme*. We implement this scheme for affine spirals. The symmetric difference distance between an affine spiral arc and its equisymmetric bitangent conic arc is a monotone function of the arc length of the spiral section. This useful property gives rise to an efficient bisection based algorithm computing the equisymmetric conic spline. The theoretical and experimental results for complexity for several curves match exactly.

Related work. McClure and Vitale [4] consider the problem of approximating a convex C^2-curve C in the plane by an inscribed n-gon with respect to the *symmetric difference metric* δ_S. They prove that, with regard to the symmetric difference distance, the optimal n-gon P_n, satisfies $\delta_S(C, P_n) = \frac{1}{12}(\int_0^l \kappa^{1/3}(s)ds)^3 \frac{1}{n^2} + O(\frac{1}{n^4})$, where κ is the Euclidean curvature of the curve C and s is the arc length parameter. Ludwig [3] shows that the symmetric difference distance of an optimal parabolic spline with n knots and a convex C^4-curve C in the plane satisfies $\delta_S(C, Q_n) = \frac{1}{240}(\int_0^l |k(u)|^{1/5}du)^5 \frac{1}{n^4} + O(\frac{1}{n^5})$, where u is the affine arc length parameter and k is the affine curvature of the curve C. Ghosh, Petitjean and Vegter [2] present the first sharp asymptotic bounds for an optimal parabolic and conic spline approximation for a sufficiently smooth curve with non-vanishing curvature, with respect to the Hausdorff distance. Furthermore, *bitangent conic arcs* of affine spirals have some useful global properties which gives rise to a simple bisection algorithm for computation of optimal conic splines.

Overview. Section 2 reviews some notions from affine differential geometry that we use in this paper. Section 3 introduces affine spirals, a class of curves which have a unique equisymmetric bitangent conic. The complexity analysis of optimal parabolic and conic splines and equisymmetric conic splines is discussed in Section 4. Section 5 presents the output of the algorithm for a specific example.

2 Mathematical preliminaries

Circular arcs and straight line segments are the only regular smooth curves in the plane with constant Euclidean curvature. Conic arcs are the only smooth curves in the plane with constant *affine curvature*. The latter property is crucial for our approach, so we briefly review some concepts and properties from affine differential geometry of planar curves. See also Blaschke [1].

Affine curvature. Recall that a regular curve $\alpha : J \to \mathbb{R}^2$ defined on a closed real interval J, i.e., a curve with non-vanishing tangent vector $T(s) := \alpha'(s)$, is parametrized according to Euclidean arc length if its tangent vector T has unit length. For a curve parametrized by arc length, the derivative of the tangent vector $N(s)$, and the Euclidean curvature is a differential invariant of regular curves under the group of rigid motions of the plane, i.e., a regular curve is uniquely determined by its Euclidean curvature, upto a rigid motion.

The larger group of *equi-affine transformations* of the plane, i.e., linear transformations with determinant one (in other words, area preserving linear transformations), also gives rise to a differential invariant, called the *affine curvature* of the curve. To introduce this invariant, let $I \subset \mathbb{R}$ be an interval, and let $\gamma : I \to \mathbb{R}^2$ be a smooth, regular plane curve. The curve γ is parametrized according to *affine arc length* if

$$[\gamma'(r), \gamma''(r)] = 1. \tag{1}$$

Here $[v, w]$ denotes the determinant of the pair of vectors $\{v, w\}$. It follows from (1) that γ has non-zero Euclidean curvature. Conversely, every curve $\alpha : J \subset \mathbb{R} \to \mathbb{R}^2$ with non-zero Euclidean curvature satisfies $[\alpha'(s), \alpha''(s)] \neq 0$, for $u \in J$, so it can be reparametrized according to affine arc length.

Note that the property of being parametrized according to affine arc length is an invariant of the curve under equi-affine transformations. If γ is parametrized according to affine arc length, then differentiation of (1) yields $[\gamma'(r), \gamma'''(r)] = 0$, so there is a scalar function k such that

$$\gamma'''(r) + k(r)\gamma'(r) = 0. \tag{2}$$

The quantity $k(r)$ is called the *affine curvature* of the curve γ at $\gamma(r)$. A regular curve is uniquely determined by its affine curvature, up to an equi-affine transformation of the plane.

The affine curvature can be expressed in terms of the derivatives of γ up to and including order four. We refer to [2] for details.

At a point of non-vanishing Euclidean curvature there is a unique conic, called the *osculating conic*, having fourth order contact with the curve at that point(or, in other words, having five coinciding points of intersection with the curve). The affine curvature of this conic is equal to the affine curvature of the curve at the point of contact. Moreover, the contact is of order five if the affine curvature has vanishing derivative at the point of contact. (The curve has to be C^5.) In that case the point of contact is a *sextactic point*. See [1] for further details.

Conics have constant affine curvature. Solving the differential equation (2) shows that a curve of constant

affine curvature is a conic arc. More precisely, a curve is a hyperbolic, parabolic or elliptic arc iff its affine curvature is negative, zero, or positive, respectively.

3 Near optimal approximation of affine spirals

We *conjecture* that there is a unique optimal bitangent conic minimizing the symmetric difference distance. Since we do not have a proof of this property yet, we introduce the *equisymmetric bitangent conic*, yielding splines that are near-optimal approximants with respect to the symmetric difference distance, and having the same asymptotic complexity as optimal conic splines.

Area function. The symmetric difference distance between a convex curve α and a chord $\alpha(\sigma)\alpha(\tau)$ is given by $A_\alpha(\sigma,\tau) = \frac{1}{2}|\int_\sigma^\tau [\alpha(u) - \alpha(\sigma), \alpha'(u)]du|$. A *bitangent conic* of a regular curve $\gamma : [0,\varrho] \to \mathbb{R}^2$, which is tangent to γ at $\gamma(0)$ and $\gamma(\varrho)$ and intersects it at $\gamma(\sigma)$ has a parametrization $\beta : [0,\varrho] \to \mathbb{R}^2$ of the form

$$\beta(r) = \gamma(r) + r^2(r-\varrho)^2(P(r,\varrho)t(r) + Q(r,\varrho)n(r)), \tag{3}$$

where $t(r) := \gamma'(r)$ is the affine tangent and $n(r) := \gamma''(r)$ is the affine normal to γ at $\gamma(r)$. The symmetric difference distance between γ and the bitangent conic β is equal to

$$\delta_S(\gamma,\beta) = |A_\gamma(0,\sigma) - A_\beta(0,\sigma)| + |A_\gamma(\sigma,\varrho) - A_\beta(\sigma,\varrho)|$$

There is a one-parameter family of bitangent conics, and the goal is to determine an *equisymmetric bitangent conic*, i.e., a conic in this family for which the area of the two moons (see Figure 1) formed by γ and β are equal. The symmetric difference distance in this case is defined to be the *equisymmetric distance* between γ and β.

Monotonicity of equisymmetric distance. If one endpoint of the affine spiral moves along the curve γ, the symmetric difference distance between the affine spiral and its equisymmetric bitangent conic arc is *monotone* in the arc length of the affine spiral. More precisely, let $\gamma : [u_0, u_1] \to \mathbb{R}^2$ be an affine spiral arc. For $u_0 \le u \le u_1$, let γ_u be the sub-arc between $\gamma(u_0)$ and $\gamma(u)$, and let β_u be the (unique) equisymmetric bitangent conic arc of γ_u. Then the equisymmetric distance between γ_u and β_u is a monotonically increasing function of u.

This property gives rise to a bisection based method for the computation of an equisymmetric conic spline approximating a spiral arc to within a given symmetric difference distance. Section 5 presents the output of this algorithm for Cayley's sextic.

4 Complexity of conic splines

In this section our goal is to determine the symmetric difference distance of a conic arc of best approximation to an arc of γ of affine arc length $\varrho > 0$, that is tangent to γ at its endpoints. If the conic is a parabola, these conditions uniquely determine a parabolic arc. If we approximate by a general conic, there is one degree of freedom left, which we use to minimize the symmetric difference distance between the arc of γ and the approximating conic arc β.

The main result of this section gives an asymptotic bound on this symmetric difference distance .

Theorem 1 (Optimal symmetric difference) *Let $\gamma : [0,\varrho] \to \mathbb{R}^2$ be a sufficiently smooth, regular curve with non-vanishing Euclidean curvature.*
1. Let β be the parabolic arc tangent to γ at the endpoints, the symmetric difference between the two arcs has the following asymptotic expansion

$$\delta_S(\gamma,\beta) = \tfrac{1}{240}|k_0|\varrho^5 + O(\varrho^6), \tag{4}$$

where k_0 is the affine curvature of γ at $\gamma(0)$.

2. Let β be a bitangent conic arc, minimizing the symmetric difference, then the symmetric difference between the two arcs has the following asymptotic expansion

$$\delta_S(\gamma,\beta) = \tfrac{1}{7680}|k_0'|\varrho^6 + O(\varrho^7), \tag{5}$$

where k_0' is the derivative of the affine curvature of γ at $\gamma(0)$.

3. Let β be the equisymmetric bitangent conic arc of γ, then the asymptotic expansion of the symmetric difference between the two curves is given by (5)

Here we just outline the main idea of the proof. Let $\gamma : [0,\varrho] \to \mathbb{R}^2$, be a curve parametrized by affine arc length. In particular ϱ is the affine arc length. Using the parametrization of β as given by (3) we have

$$\delta_S(\gamma,\beta) = \tfrac{1}{30}|Q(0,0)|\varrho^5 + O(\varrho^6). \tag{6}$$

In [2], we show that the affine curvature of a curve of the form (3) is given by

$$k_\beta = k_0 + 8Q(0,0) + O(\varrho). \tag{7}$$

Since β is a parabolic arc, its affine curvature is zero, i.e., $k_\beta = 0$. Combining (3), (6), and (7) yields the asymptotic expression for the symmetric difference distance given by (4). The proof of the second and third part follows the same line of reasoning.

Corollary 2 (Complexity of conic spline) *Let $\gamma : [0,\varrho] \to \mathbb{R}^2$ be a smooth curve with non-vanishing Euclidean curvature, parametrized by affine arc*

length, and let $k(r)$ be its affine curvature at $\gamma(r)$.
1. The minimal number of arcs in a tangent continuous parabolic spline approximating γ to within symmetric difference distance ε is

$$N(\varepsilon) = (240)^{-1/4}\Big(\int_0^{\varrho} |k(r)|^{1/5} dr\Big)\varepsilon^{-1/4}(1 + O(\varepsilon^{1/4})).$$

2. The minimal number of arcs in a tangent continuous conic spline approximating γ, to within symmetric difference distance ε is

$$N(\varepsilon) = (7680)^{-1/5}\Big(\int_0^{\varrho} |k'(r)|^{1/6} dr\Big)\varepsilon^{-1/5}(1+O(\varepsilon^{1/5})).$$

The expression for complexity of an equisymmetric conic spline is of the same form as the expression for complexity of an optimal conic spline as given in 2. The expressions match in the most significant terms. For all practical cases this difference turned out to be negligible.

Remark. The basic idea behind proving the preceding corollary is to define the functions called *parabolic content* and *conic content*. These functions are useful in distributing the knots over the curve γ, in such a way, that the symmetric difference distance of all the segments are equal. The aim for this kind of approximation is to distribute the knots uniformly over the curve with respect to the parabolic or conic content. In fact the methods used by McClure and Vitale in [4] and Ludwig in [3] use this notion of content to show that there exists an optimal spline minimizing the symmetric difference distance for a curve with a given number of knots.

Note that $N(\varepsilon)$ is expressed in terms of equi-affine invariants, affine curvature and affine arc length, since the symmetric difference metric is invariant under equi-affine transformations.

5 Implementation

We implemented an algorithm in `C++` using the symbolic computing library `GiNaC`[1], for the computation of an optimal parabolic or an equisymmetric conic spline, based on the monotonicity property. For computing the optimal parabolic spline, the curve is divided into affine spirals at the sextactic points. Then for a local stopping condition ε_l, the algorithm iteratively computes the optimal parabolic arcs starting at one endpoint. Given symmetric difference distance ε we compute ε_l, by first computing the complexity n from our theoretical result, where $\varepsilon_l = \frac{\varepsilon}{n}$. Infact our algorithm gives an exact match between the theoretical complexity and the experimental complexity, for sufficiently small values of ε.

[1] http://www.ginac.de

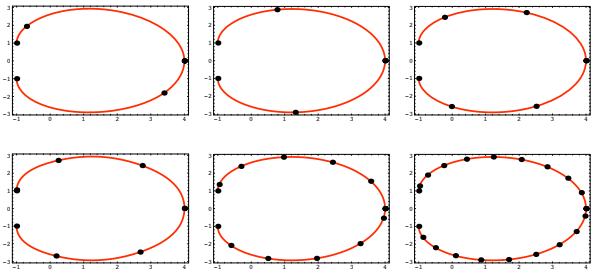


Figure 2: Row 1 shows the conic spline and row 2 shows parabolic spline approximation for Cayley's sextic for $\varepsilon = 10^{-1}, 10^{-2}$ and 10^{-3}

Cayley's sextic. We present the results of our algorithm applied to *Cayley's sextic*, parametrized by $\alpha(t) = (4\cos(\frac{t}{3})^3\cos(t), 4\cos(\frac{t}{3})^3\sin(t)$, with $-\frac{3}{4}\pi \leq t \leq \frac{3}{4}\pi$. This curve has a sextactic point at $t = 0$, therefore for all values of ε we divide the parameter interval into two parts $[-\frac{3}{4}\pi, 0]$ and $[0, \frac{3}{4}\pi]$.

Table 1 gives the number of arcs computed by the algorithm, and the theoretical bounds on the number of arcs for varying values of ε, both for the parabolic and for the conic spline.

ε	Parabolic Exp.	Th.	Conic Exp.	Th.
10^{-1}	6		4	
10^{-2}	12		4	
10^{-3}	20		6	
10^{-4}	34		10	
10^{-5}	60		16	
10^{-6}	108		24	

Table 1: Theoretical and experimental complexity match exactly for parabolic and conic spline approximation of Cayley's sextic for various values of symmetric difference distance ε

References

[1] W. Blaschke. *Vorlesungen über Differentialgeometrie II. Affine Differential Geometrie*, volume VII of *Die Grundlehre der mathematischen Wissenschaften in Einzeldarstellungen*. Springer-Verlag, 1923.

[2] S. Ghosh, S. Petitjean, and G. Vegter. Approximation by conic splines. *Mathematics in Computer Science*, 1:36–69, 2007.

[3] M. Ludwig. Asymptotic approximation by quadratic spline curves. *Annales Universitatis Scientiarum Budapestinensis. Sectio Mathematica*, 42:133–139, 1999.

[4] D.E. McClure and R. A. Vitale. Polygonal approximation of plane convex bodies. *Journal of Mathematical Analysis and Applications*, 51:326–358, 1975.

[5] L. Fejes Tóth. Approximations by polygons and polyhedra. *Bull. Amer. Math. Soc.*, 54:431–438, 1948.

Mixed Volume Techniques for Embeddings of Laman Graphs

Reinhard Steffens* Thorsten Theobald*

Abstract

We use Bernstein's Theorem [1] to obtain combinatorial bounds for the number of embeddings of Laman graph frameworks modulo rigid motions. For this, we study the mixed volume of suitable systems of polynomial equations obtained from the edge length constraints. The bounds can easily be computed and for some classes of graphs, the bounds are tight.

1 Introduction

Let $G = (V, E)$ be a graph with $|E| = 2|V| - 3$ edges. If each subset of k vertices spans at most $2k-3$ edges, we say that G has the *Laman property* and call it a *Laman graph* (see [7]). For generic edge lengths, Laman graphs are minimally rigid (see [3]), i.e. they become flexible if any edge is removed.

A *Henneberg sequence* for a graph G is a sequence $(G_i)_{3\leq i\leq n}$ of Laman graphs such that G_3 is a triangle, $G_n = G$ and each G_i is obtained by G_{i-1} via one of the following two types of steps: A *Henneberg I step* adds one new vertex v_{i+1} and two new edges, connecting v_{i+1} to two arbitrary vertices of G_i. A *Henneberg II step* adds one new vertex v_{i+1} and three new edges, connecting v_{i+1} to three vertices of G_i such that at least two of these vertices are connected via an edge e of G_i and this certain edge e is removed (see Figure 1). Any Laman graph G can be constructed via a

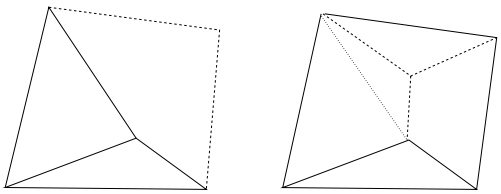


Figure 1: A Henneberg I and a Henneberg II step. New edges are dashed and the deleted edge is pointed.

Henneberg sequence and any graph constructed via a Henneberg sequence has the Laman property (see [9]). We call G a *Henneberg I graph* if it is constructable using only Henneberg I steps. Otherwise we call it *Henneberg II*.

In the following we look at *frameworks* which are tuples (G, L) where $G = (V, E)$ is a graph and $L = \{l_{i,j} : [v_i, v_j] \in E\}$ is a set of $|E|$ positive numbers interpreted as edge lengths. Given a framework we want to know how many embeddings, i.e. maps $\alpha : V \to \mathbb{R}^2$, exist such that the Euclidean distance between two points in the image is exactly $l_{i,j}$ for all $[v_i, v_j] \in E$. Since every rotation or translation of an embedding gives another one, we ask how many embeddings exist *modulo rigid motions*.

Due to the minimal rigidity property, questions about embeddings of Laman graphs arise naturally in rigidity and linkage problems (see [2] and the references therein). Graphs with fewer edges will have zero or infinitely many embeddings modulo rigid motions, and graphs with more edges do not have any embeddings for a generic choice of edge lengths.

Determining the maximal number of embeddings (modulo rigid motions) for a given Laman graph is an open problem. The best upper bounds are due to Borcea and Streinu [2] who show that the number of embeddings is bounded by $\binom{2|V|-4}{|V|-2}$. Their bounds are based on degree results of determinantal varieties, but do not seem to fully exploit the specific combinatorial structure of Laman graphs.

Here, we present an alternative, combinatorial approach to bound the number of embeddings of a Laman graph based on Bernstein's theorem for sparse polynomial systems. Since the systems of polynomial equations describing the Laman embeddings are sparse, the mixed volume of the Newton polytopes provides a simple combinatorial upper bound on the number of solutions. It is particularly interesting that for some classes of graphs, the mixed volume bound is tight (and in these cases improves the general bound in [2]).

To use algebraic tools for this problem we formulate the embedding problem as a system of polynomial equations. Each prescribed edge length translates into a polynomial equation. I.e. if $[v_i, v_j] \in E$ with length $l_{i,j}$, we require $(x_i - x_j)^2 + (y_i - y_j)^2 = l_{i,j}^2$ where $\alpha(v_i) = (x_i, y_i)$ and $\alpha(v_j) = (x_j, y_j)$. Thus we obtain a system of $|E|$ quadratic equations whose solutions represent the embeddings of our framework. To get rid of translations and rotations we fix one point $\alpha(v_1) = (x_1, y_1) = (c_1, c_2)$ and the direction of the edge $[\alpha(v_1), \alpha(v_2)]$ by setting $y_2 = c_3$. (Here we assume without loss of generality that there is an edge between v_1 and v_2.) For practical reasons we choose $c_i \neq 0$ and as well $c_1 \neq l_{1,2}$. Hence we want to study

*FB 12 – Institut für Mathematik, Postfach 111932, D–60054 Frankfurt am Main, Germany

the solutions to the following system.

$$\left\{\begin{array}{l} x_1 - c_1 = 0 \\ y_1 - c_2 = 0 \\ x_2 - (l_{1,2} - c_1) = 0 \\ y_2 - c_3 = 0 \\ (x_i - x_j)^2 + (y_i - y_j)^2 - l_{i,j}^2 = 0 \\ \qquad \forall [v_i, v_j] \in E - \{[v_1, v_2]\} \end{array}\right\} \quad (1)$$

We will give bounds on the number of solutions in $\mathbb{C}^* := \mathbb{C} \setminus \{0\}$ to this system where we assume that the edge lengths $l_{i,j}$ are generically chosen such that no solutions with zero components occur. To do this we will study the mixed volume of the Newton polytopes (i.e. the convex hulls of the monomial exponent vectors, see for example [8]) of the system (1).

2 Bernstein's Theorem and technical tools

Let $P_1, \ldots, P_n$ be n polytopes in $\mathbb{R}^n$. For non-negative parameters $\lambda_1, \ldots, \lambda_n$, the volume $\mathrm{vol}_n(\lambda_1 P_1 + \ldots + \lambda_n P_n)$ is a homogeneous polynomial of degree n in $\lambda_1, \ldots, \lambda_n$ with non-negative coefficients (see [10]). The coefficient of the monomial $\lambda_1 \cdots \lambda_n$ is called the *mixed volume of* $P_1, \ldots, P_n$ and is denoted by $\mathrm{MV}_n(P_1, \ldots, P_n)$. We have two explicit formulas for this quantity (see [8] and [5]):

$$\begin{aligned} &\mathrm{MV}_n(P_1, \ldots, P_n) \\ &\quad = (-1)^n \sum_{(\alpha_1,\ldots,\alpha_n)\in\{0,1\}^n} (-1)^{\sum_i \alpha_i} \mathrm{vol}_n\left(\sum_i \alpha_i P_i\right) \quad (2) \\ &\quad = \sum_{\substack{Q \text{ mixed cell of a} \\ \text{mixed subdivision} \\ \text{of } P := \sum P_j}} \mathrm{vol}_n(Q) \quad (3) \end{aligned}$$

For further background on mixed subdivisions, see also [5] and [4].

The core theorem that gives a connection between solutions to systems of polynomial equations and discrete geometry is the following.

Theorem 1 (Bernstein [1]) *Given polynomials $f_1, \ldots, f_n$ over $\mathbb{C}$ with finitely many common zeroes in $(\mathbb{C}^*)^n$, let P_i denote the Newton polytope of f_i in $\mathbb{R}^n$. Then the number of common zeroes of the f_i in $(\mathbb{C}^*)^n$ is bounded above by the mixed volume $MV_n(P_1, \ldots, P_n)$. Moreover for generic choices of the coefficients in the f_i, the number of common solutions is exactly $MV_n(P_1, \ldots, P_n)$.*

Bernstein also gives an explicit condition when a choice of coefficients is generic. We can show that the system (1) is never generic in that sense. Then the mixed volume of it will always be a strict upper bound on the number of common solutions.

In the special case of Henneberg I graphs our system (1) will be in a shape that allows to separate the mixed volume calculation into smaller pieces. Our main tool to do this is the following Lemma.

Lemma 2 *Let $P_1, \ldots, P_k$ be polytopes in $\mathbb{R}^{m+k}$ and $Q_1, \ldots, Q_m$ be polytopes in $\mathbb{R}^m \subset \mathbb{R}^{m+k}$. Then*

$$\begin{aligned} &\mathrm{MV}_{m+k}(Q_1, \ldots, Q_m, P_1, \ldots, P_k) = \\ &\qquad \mathrm{MV}_m(Q_1, \ldots, Q_m) * \mathrm{MV}_k(\pi(P_1), \ldots, \pi(P_k)) \end{aligned}$$

where $\pi : \mathbb{R}^{m+k} \to \mathbb{R}^k$ denotes the projection on the last k coordinates.

Proof. Using the explicit formula (2) we have:

$$\begin{aligned} &\mathrm{MV}_{m+k}(Q_1, \ldots, Q_m, P_1, \ldots, P_k) \\ &= (-1)^{m+k} \sum_{\beta\in\{0,1\}^k} \sum_{\alpha\in\{0,1\}^m} (-1)^{\sum_i \alpha_i} (-1)^{\sum_j \beta_j} \\ &\qquad \mathrm{vol}_{m+k}\left(\sum_{i=1}^m \alpha_i Q_i + \sum_{j=1}^k \beta_j P_j\right) . \end{aligned}$$

Since any polytopes $P \subset \mathbb{R}^{m+k}$ and $Q \subset \mathbb{R}^m$ satisfy $\mathrm{vol}_{m+k}(Q + P) = \mathrm{vol}_m(Q)\ \mathrm{vol}_k(\pi(P)) + \mathrm{vol}_{m+k}(P)$, this equals

$$\begin{aligned} &(-1)^{m+k} \sum_{\beta\in\{0,1\}^k} \sum_{\alpha\in\{0,1\}^m} (-1)^{\sum_i \alpha_i \sum_j \beta_j} \\ &\quad * \left[\mathrm{vol}_m\left(\sum_{i=1}^m \alpha_i Q_i\right) \mathrm{vol}_k\left(\pi(\sum_{j=1}^k \beta_j P_j)\right)\right. \\ &\qquad \left. + \mathrm{vol}_{m+k}\left(\sum_{j=1}^k \beta_j P_j\right)\right] . \end{aligned}$$

Using that $\pi(P_1 + P_2) = \pi(P_1) + \pi(P_2)$ for any polytopes $P_1, P_2 \subset \mathbb{R}^{m+k}$, we obtain

$$\begin{aligned} &(-1)^{m+k} \sum_{\alpha\in\{0,1\}^m} (-1)^{\sum_i \alpha_i} \left[\sum_{\beta\in\{0,1\}^k} (-1)^{\sum_j \beta_j}\right. \\ &\qquad \left. * \mathrm{vol}_{m+k}\left(\sum_{j=1}^k \beta_j P_j\right)\right] \\ &+ (-1)^m \sum_{\alpha\in\{0,1\}^m} (-1)^{\sum_i \alpha_i} \mathrm{vol}_m\left(\sum_{i=1}^m \alpha_i Q_i\right) \\ &* \left[(-1)^k \sum_{\beta\in\{0,1\}^k} (-1)^{\sum_j \beta_j} \mathrm{vol}_k\left(\pi(\sum_{j=1}^k \beta_j P_j)\right)\right] . \end{aligned}$$

Now the first two lines equal 0 because we just add and substract 2^{m-1} times the term in square brackets, the third line is $\mathrm{MV}_m(Q_1, \ldots, Q_m)$ and finally the last line equals $\mathrm{MV}_k(\pi(P_1), \ldots, \pi(P_k))$ according to our alternating formula for the mixed volume (2). □

Another technical tool which will be needed in a subsequent proof is the following Lemma. This goes back to an idea of Emiris and Verschelde [4] to use linear programming and the formula (3) to compute the mixed volume. The proof (which we do not give here) is based on the duality theorem for linear programming.

Lemma 3 *Given polytopes $P_1, \ldots, P_n \subset \mathbb{R}^n$ and lifting vectors $\mu_1, \ldots, \mu_n \in \mathbb{R}^n_{\geq 0}$. Denote the vertices of P_i by $v_1^{(i)}, \ldots, v_{m_i}^{(i)}$ and choose one edge $e_i = [v_{k_i}^{(i)}, v_{l_i}^{(i)}]$ from each P_i. Then $\sum_{i=1}^n e_i$ is a mixed cell of the mixed subdivision induced by the liftings μ_i if and only if*

i) The edge matrix $E := V_a - V_b$ is non-singular (where $V_a := (v_{k_1}^{(1)}, \ldots, v_{k_n}^{(n)})$ and $V_b := (v_{l_1}^{(1)}, \ldots, v_{l_n}^{(n)})$) and

ii) For all polytopes P_i and all vertices $v_s^{(i)}$ of P_i which are not in e_i we have:

$$\left(\operatorname{diag}\left(\mu^T E\right)^T E^{-1} - \mu_i^T\right) \cdot \left(v_{l_i}^{(i)} - v_s^{(i)}\right) \geq 0 \quad (4)$$

where $\mu := (\mu_1, \ldots, \mu_n)$ and where $\operatorname{diag}(V)$ denotes the vector of the diagonal entries of V.

Note that (4) is linear in the μ_j. Hence given a choice of edges we can explicitly calculate $\sum_{i=1}^n m_i$ normal vectors defining a cone in $\mathbb{R}^{n^2}$. The interior of this cone consists of all liftings $(\mu_1^t, \ldots, \mu_n^t)$ which induce a mixed subdivision that contains our chosen cell as a mixed cell.

3 Henneberg I graphs

For this simple class of Laman graphs the mixed volume bound is tight as we will demonstrate below. Our proof exploits the inductive structure of Henneberg I graphs which is why it cannot be used for Henneberg II graphs.

Theorem 4 *A Henneberg I step at most doubles the number of embeddings of the framework and there is always a choice of edge lengths such that the number of embeddings is doubled.*

Proof. In a Henneberg I step we add one vertex $v_{|V|+1}$ and two edges $[v_r, v_{|V|+1}], [v_s, v_{|V|+1}]$ with lengths $l_{r,|V|+1}$ and $l_{s,|V|+1}$. So our system of equations (1) gets two new equations, namely

$$(x_r - x_{|V|+1})^2 + (y_r - y_{|V|+1})^2 - l_{r,|V|+1}^2 = 0 \quad (5)$$
$$(x_s - x_{|V|+1})^2 + (y_s - y_{|V|+1})^2 - l_{s,|V|+1}^2 = 0. \quad (6)$$

In our new system of equations these two are the only polynomials involving $x_{|V|+1}$ and $y_{|V|+1}$, so we can use Lemma 2 to calculate the mixed volume separately. Unfortunately, the mixed volume of the projection of the Newton polytopes of these equations equals 4 which would imply that the number of embeddings is at most quadrupled. But the following simple trick (which we will refer to as the *truncation trick*) solves this problem immediately. The set of solutions of a system of polynomial equations is not changed when we substract one equation from another. So instead of adding equation (6) we add the equation (6)-(5) which equals

$$\begin{aligned} x_s^2 - x_r^2 + 2x_{|V|+1}(x_r - x_s) + y_s^2 - y_r^2 + \\ 2y_{|V|+1}(y_r - y_s) - l_{s,|V|+1}^2 + l_{r,|V|+1}^2 = 0 \,. \end{aligned} \quad (7)$$

Now the projections of the two new Newton polytopes corresponding to (5) and (7) to their last two coordinates have mixed volume 2 which proves the first part of our theorem. To get two new embeddings for each previous one we choose our new edge lengths to be almost equal to each other and much larger then all previous edges lengths (larger then the sum of all previous is certainly enough). This leads to the desired new embeddings. □

Each Henneberg sequence starts with a triangle which has obviously at most 2 embeddings up to rigid motions (we count reflections separately). Hence using our Theorem inductively we get the following corollary.

Corollary 5 *The number of embeddings of Henneberg I graphs is less than or equal $2^{|V|-2}$ and this bound is sharp.*

4 Laman graphs on 6 vertices

For Laman graphs on 6 vertices, the general bound in [2] on the number of embeddings is 70. From the Henneberg constructions and simple combinatorial considerations, it follows that the only Henneberg II Laman graphs on 6 vertices are the Desargues graph and $K_{3,3}$ (see figure 2). For the Desargues graph, an

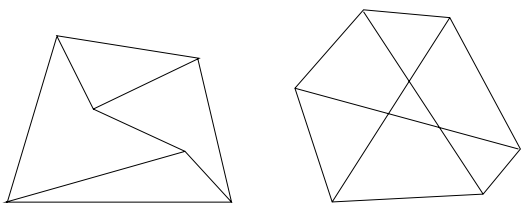


Figure 2: Left: Desargues graph. Right: $K_{3,3}$.

explicit analysis is given in [2] which shows that the correct number is only 24, and that there is a choice of edge lengths giving 24 different embeddings. For

the $K_{3,3}$, Manfred Husty found a construction with 32 embeddings [6].

When we set up the system (1) and use the truncation trick like in the proof of Theorem 4 several times, our mixed volume approach yields a bound of 32 for both graph classes on 6 vertices. So in the case of 6 vertices our bound is tight. By glueing several copies of $K_{3,3}$ together and using Lemma 2 to calculate the mixed volume we get an infinite class of graphs where our bound is tight as well.

5 General case

For the classes discussed above (Henneberg I, graphs on six vertices) as well as some other special cases, our bound on the number of embeddings improves the known general bounds. We were not able to generalize the truncation trick to arbitrary Henneberg II graphs. For the general case, our mixed volume approach for the untruncated system (1) provides a simple, but very weak bound. However, it may be of independent interest, that for this class of problems, it is possible to determine the mixed volume exactly.

Theorem 6 *The mixed volume of our initial system (1) is exactly* $4^{|V|-2}$.

Proof. The mixed volume of (1) is at most the product of the degrees of the polynomial equations because it is less than or equal to the Bézout bound (see [8]). To show that the mixed volume is at least this number we will use Lemma 3 to give a lifting that induces a mixed cell of volume $4^{|V|-2}$.

The first 4 equations of (1) give rise to a single edge as a Newton polytope which is part of any mixed cell. Now we claim that we can order the Newton polytopes P_i in such a way that, for $i \geq 5$, P_i contains the edge $[0, 2\xi_i]$ where ξ_i denotes the i^{th} unit vector. To see this, note first that every equation in (1) has a non vanishing constant term and therefore its Newton polytope contains the point 0. To see that P_i contains $2\xi_i$ it is enough to show there is a labeling of the edges of our graph with a direction such that each vertex has exactly two incoming edges. Figure 3 sketches how to choose the edge directions in the Henneberg steps to satisfy this. Now using Lemma 3 we describe a lifting that induces a subdivision that has $\sum_i [0, 2\xi_i]$ as a mixed cell. In the notation of Lemma 3 our chosen edges give rise to the edge matrix $E = \begin{pmatrix} \mathbb{E}_4 & \mathbf{0} \\ \mathbf{0} & 2\mathbb{E}_{2|V|-4} \end{pmatrix}$. Substituting this into the second condition (4) we get that for each Newton polytope P_i all vertices $v_s^{(i)}$ of P_i which are not 0 or $2\xi_i$ have to satisfy

$$\left((\mu_1^{(1)}, \ldots, \mu_{2|V|}^{(2|V|)}) - \mu^{(i)}\right) \cdot v_s^{(i)} \leq 0\,,$$

where we denote by $\mu^{(j)} \in \mathbb{Q}^{2|V|}$ the lifting vector for P_j. Since all the entries of each $v_s^{(i)}$ are non-negative this can easily be done by choosing the vectors $\mu^{(j)}$ such that their j^{th} entry is relatively small and all other entries are relatively large. □

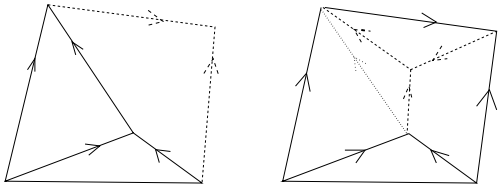


Figure 3: A Henneberg I and a Henneberg II step with directed edges.

Corollary 7 *The number of embeddings of a Laman graph framework with generic edge lengths is strictly less then* $4^{|V|-2}$.

References

[1] D.N. Bernstein. The number of roots of a system of equations. *Funkcional. Anal. i Priložen.*,9(3):1-4, 1975.

[2] C. Borcea and I. Streinu. The number of embeddings of minimally rigid graphs, *Discrete Comput. Geom.*, 31(2):287-303, 2004.

[3] R. Conelly. Rigidity. In *Handbook of Convex Geometry, Vol. A*, pages 223-271. North Holland, Amsterdam, 1993.

[4] I.Z. Emiris and J. Verschelde. How to count efficiently all affine roots of a polynomial system. *Discrete Appl. Math.*, 93(1):21-32, 1999.

[5] B. Huber and B. Sturmfels. A polyhedral method for solving sparse polynomial systems. *Math. Comp.*, 64(212):1541-1555, 1995.

[6] M. Husty. Talk given at the IMA workshop 'Applications in Biology, Dynamics and Statistics', May 2007.

[7] G. Laman. On graphs and rigidity of plane skeletal structures. *J. Engrg. Math.*, 4:331-340, 1970.

[8] B. Sturmfels. *Solving systems of polynomial equations*, volume 97 of *CBMS Regional Conference Series in Mathematics*, 2002.

[9] T.-S. Tay and Walter Whiteley. Generating isostatic graphs. *Structural Topology*, (11):21-69, 1985.

[10] R. Webster. *Convexity*, Oxford University Press, New York, 1994.

Geometric Analysis of Algebraic Surfaces Based on Planar Arrangements

Eric Berberich* Michael Kerber* Michael Sagraloff*

Abstract

We present a method to compute the exact topology of a real algebraic surface S, implicitly given by a polynomial $f \in \mathbb{Q}[x, y, z]$ of arbitrary degree N. Additionally, our analysis provides geometric information as it supports the computation of arbitrary precise samples of S including critical points. We use a projection approach, similar to Collins' cylindrical algebraic decomposition (cad). In comparison we reduce the number of output cells to $O(N^5)$ by constructing a special planar arrangement instead of a full cad in the projection plane. Furthermore, our approach applies numerical and combinatorial methods to minimize costly symbolic computations. The algorithm handles all sorts of degeneracies without transforming the surface into a generic position. We provide a complete C++-implementation of the algorithm that shows good performance for many well-known examples from algebraic geometry.

1 Introduction

Problem and results: The topological analysis of algebraic curves and surfaces has received a lot of attention in algebraic geometry, computer graphics and CAGD. Beside the theoretical interest of the problem, accurate topological and geometric information of algebraic objects is crucial for a good visualization and for a meaningful approximation by simpler objects, such as splines or polygons.

We present an algorithm that provides topological information about an arbitrary algebraic surface S, given by an implicit equation in $\mathbb{Q}[x, y, z]$ of degree N. We compute a cell decomposition, where each cell is a smooth subvariety of S of dimension 0, 1, or 2, and determine how these cells are connected. Our cell decomposition has the *boundary property*, i.e., the boundary of a cell is given by a union of other cells (compare the similar notion of a CW-complex from algebraic topology). The result is similar to a *cylindrical algebraic decomposition* of $\mathbb{R}^3$, but our decomposition represents the topology using only $O(N^5)$ cells whereas the worst case complexity of a cad is $\Omega(N^7)$.

Our algorithm consists of three steps: First, we *project* the z-critical points of S to compute an arrangement $\mathcal{A}_S$, see Section 2. Second, we *lift* the components of $\mathcal{A}_S$ to $\mathbb{R}^3$, obtaining the cell decomposition Ω_S. It suffices to lift over one sample point of each component. Details are in Section 3. Third, we compute the *adjacencies* between the cells of Ω_S, as explained in Section 4.

We describe new methods for all three steps with the goal to replace costly symbolic computations by certified approximation methods as much as possible. Our toolbox for approximate methods contains, for instance, a numerical method for univariate root isolation (Bitstream Descartes [8]), an extension for the non-square-free case (m-k-Bitstream Descartes [7]), and interval arithmetic. Still, we guarantee to reflect the mathematical correct topology of the surface in all cases, as expected from the *exact geometric computation* (EGC) paradigm.

Our approach does not make any assumptions about the input surface and does never transform the coordinate system to prevent degeneracies. This allows to accurately sample the surface in arbitrary resolution by lifting points of a fine granulation of the xy-plane. On the other hand, we have to deal with degenerate situations, in particular with vertical lines that are part of the surface. Such lines are decomposed into vertical segments, and vertices in-between, to satisfy the boundary property.

We also provide an exact and complete implementation of the presented algorithm in C++. To our knowledge, this is the first EGC-implementation for the topological analysis of algebraic surfaces, including singular ones. It relies on an EGC-algorithm to produce arrangements of arbitrary algebraic plane curves, which has been presented recently in [6]. Our experiments show good performance for reference surfaces from algebraic geometry. Essentially needed in the projection step of our approach is the analysis of planar curves of degree up to $N(N-1)$ which limits its practical applicability for high-degree surfaces.

Related work: The problem of topology computation for algebraic plane curves has been extensively studied (see [7], [5] and the references therein). Recently, also exact methods for the case of space curves and surfaces came under consideration. Mourrain and Técourt [10] compute the topology of a surface by an isotopic piecewise linear mesh, using a plane-sweep approach. Cheng et al. [4] use a projection approach to produce a curvilinear wireframe that represents the surface topology. Both methods require a generic position of the surface and apply a linear change of coor-

*Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany, email: {eric,mkerber,msagralo}@mpi-inf.mpg.de

dinates otherwise. None of them reports on practical performance of their techniques.

Arnon et al. [1] compute cads in $\mathbb{R}^n$. In [2], they also compute the adjacencies between cells in a three-dimensional cad. Similar to us, they do not switch to generic position, and partition vertical lines into several cells to satisfy the boundary property. Our algorithm uses a more suitable cell decomposition for topology information, and applies approximate methods in the adjacency computation.

A more detailed version of this work appears in [3].

2 (n,k)-Arrangements

Throughout the article, the surface S is implicitly given by the polynomial $f \in \mathbb{Q}[x, y, z]$ of degree N. We require f to be square-free and primitive, i.e., S contains no component twice, and has no two-dimensional vertical component. For simplicity, we first assume that S contains no vertical line. The end of Section 4 sketches how to handle vertical lines.

For a fixed (algebraic) point $p = (p_x, p_y) \in \mathbb{R}^2$ we consider the *local polynomial* $f_p := f(p_x, p_y, z) \in \mathbb{R}[z]$.

Definition 1 *The* local degree n_p *is the degree of* f_p *in* z *and* $k_p := \deg \gcd(f_p, f_p')$ *the* local gcd degree.

We partition the (x, y)-plane into connected regions where the local degree and the local gcd degree remain invariant. To represent this partition, we use a planar arrangement. Thus we define:

Definition 2 *A connected set* $C \subset \mathbb{R}^2$ *is called (n,k)-invariant with respect to* S *if the local degree* $n = n_C$ *and the local gcd degree* $k = k_C$ *of* f *are invariant for all* $p \in C$. *An (n,k)-arrangement* $\mathcal{A}_S$ *for* S *is a planar arrangement whose vertices, edges, and faces are (n,k)-invariant with respect to* S.

Theorem 1 *There exists an (n,k)-arrangement* $\mathcal{A}_S$.

Proof. We give a constructive proof. Let p be an arbitrary point in the plane, and $f = \sum_{i=0}^{N} a_i(x, y)z^i$. Then n_p depends on the coefficients $a_N, \ldots, a_0$ by

$$n_p = \deg f_p = \max_{i=0,\ldots,N} \{i \mid a_i(p) \neq 0\}.$$

The same way, the local degree depends on the *principal Sturm-Habicht coefficients* $\mathrm{stha}_i(f_{n_p})$ (compare [9]) by

$$k_p = \deg \gcd(f_p, f_p') = \min_{i=0,\ldots,N} \{i \mid \mathrm{stha}_i(f_{n_p})(p) \neq 0\}.$$

The coefficients a_i's and $\mathrm{stha}_i(f_{n_p})$ define plane curves $\alpha_i = V(a_i)$ and $\sigma_{n_p,i} = V(\mathrm{stha}_i(f_{n_p}))$, respectively, of degree at most $N(N-1)$. Then n_p and k_p are determined by the curves p is part of. Thus, the arrangement $\mathcal{A}_S$ induced by α_i and, for all $n = 1, \ldots, N$, $\sigma_0(f_n), \ldots, \sigma_n(f_n)$, has only (n,k)-invariant cells. □

The constructed arrangement has (n,k)-invariant cells, but it contains far too many cells. To reduce the number of cells, consider the *silhouette* Γ_S of S, defined by $\mathrm{stha}_0(f) = \mathrm{res}_z(f, \frac{\partial f}{\partial z})$.

Lemma 2 *For any point,* $(n_p, k_p) = (N, 0)$ *if and only if* p *is not on* Γ_S. *As a consequence, all edges and vertices of an (n,k)-arrangement* $\mathcal{A}_S$ *away from* Γ_S *can be merged with their adjacent faces to an (n,k)-invariant face.*

Proof. We have that $\mathrm{res}_z(f, \frac{\partial f}{\partial z}) = a_N \mathrm{Disc}(f)$ where $\mathrm{Disc}(f)$ denotes the discriminant of f. Clearly, $n_p = N$ for a point p if and only if $a_N(p) \neq 0$. From the definition of the discriminant, $k_p = 0$ for a point p if and only if $\mathrm{Disc}(f)(p) \neq 0$. □

Consequently, any (n,k)-arrangement $\mathcal{A}_S$ can be turned into the minimal (n,k)-arrangement by a post-processing step (each feature $C \in \mathcal{A}_S$ stores (n_C, k_C) as data): Remove all edges and vertices away from Γ_S, and remove vertices on Γ_S that have exactly two adjacent edges, and both edges have the same local degree and local gcd degree as the vertex (and merge the adjacent edges). In our implementation, we integrated this post-processing step in the arrangement computation of the curves defined in the proof of Theorem 1, i.e., we add the curves into the arrangement one by one, and throw away unnecessary features immediately. This lowers the size of the intermediate values in the algorithm. One can prove that the size of the computed minimal $\mathcal{A}_S$ is $O(N^4)$, i.e., of same magnitude as the size of the silhouette arrangement.

3 The cell decomposition

We now fix an (n,k)-invariant cell C, and consider the surface lifted over C. We define the *local real degree* m_p to be the number of distinct real roots of the local polynomial f_p.

Theorem 3 *Each* $p \in C$ *has the same local real degree* m_C. *Moreover, for each* $i \in \{1, \ldots, m_C\}$, *the* i*-th lift* $C^{(i)}$ *over* C *is connected, where*

$$C^{(i)} := \{(p_x, p_y, z_i) \in \mathbb{R}^3 \mid (p_x, p_y) \in C \text{ and } z_i \text{ is the } i\text{-th root of } f_p\}.$$

Proof idea. Over an (n,k)-invariant set, the number of complex roots is constantly $n - k$. The roots of $f(p, z)$ continuously depend on p, thus in an open neighborhood of any point on C the imaginary roots stay imaginary. As the number of roots is preserved and imaginary roots only appear with their complex conjugate, the real roots also remain real. □

Theorem 3 shows that over C, the surface simply consists of m_C covertical copies of C. We can define:

Definition 3 *Let $\mathcal{A}_S$ be the minimal (n,k)-arrangement for S and m_C the local real degree of a cell $C \in \mathcal{A}_S$. The* cell decomposition *is defined as*

$$\Omega_S := \bigcup_{C \in \mathcal{A}_s} \left(\bigcup_{i=1,\ldots,m_C} \{C^{(i)}\} \right)$$

Theorem 4 *Ω_S consists of $O(N^5)$ cells.*

By computing the adjacencies between these cells as presented in Section 4, we thus can compute the topology of the surface using $O(N^5)$ many sample points which improves the $O(N^6)$ bound from [10]. A cylindrical algebraic decomposition consists of $\Omega(N^7)$ cells in the worst case, due to its vertical decomposition strategy in the plane.

The question remains how we compute the number m_C for each feature of $\mathcal{A}_S$. As we need also geometric information over C for the adjacency computation, we consider a more general problem: given $p \in C$, isolate the real roots of the local polynomial f_p. The number of isolating intervals for a sample point immediately reveals m_C.

For the isolation, we first consider the local gcd degree $k_p = k_C$ (Definition 1): if it is zero, the local polynomial is square-free, and we apply the *Bitstream Descartes method* [8], an exact root solver with adaptive precision, on f_p. Otherwise, we compute m_p, the number of real roots, using the Sturm-Habicht sequence of f_p [9], and apply the *m-k-Bitstream Descartes method* [7], an extension of the Bitstream Descartes for multiple roots, on f_p. If this steps fails (in this case, the m-k-Bitstream Descartes quits with a failure), the square-free part of f_p is computed, again using the Sturm-Habicht sequence, and the real roots of the square-free part are computed using the Bistream Descartes method.

4 Adjacency

The last step is to compute how the cells of Ω_S are connected. We first state without proof:

Theorem 5 *Ω_S has the boundary property, i.e., the boundary of each cell is the union of other cells.*

Equivalently, for any two cells M_1, M_2 with $\dim M_1 < \dim M_2$, either M_1 does not intersect the boundary of M_2, or it is completely contained in the boundary. In the latter case we call M_1 and M_2 *adjacent*. The adjacency relation of such a pair can be checked at an arbitrary point $p \in M_1$, i.e., the two cells are adjacent if and only if $p \in \overline{M_2}$.

Our strategy to compute the adjacencies is to consider all pairs of adjacent features C_1, C_2 of $\mathcal{A}_S$ in the (x, y)-plane, and to find the adjacencies between the lifts $C_1^{(i)}$ and $C_2^{(j)}$. Assume $\dim C_1 < \dim C_2$. There are two cases to consider:

C_1 has dimension 1: This means that $E := C_1$ is an edge, and $F := C_2$ is a face. As a filter, if E has at most one multiple real root, we adopt the combinatorial adjacency algorithm for plane curves from [7].

If the filter does not apply, the treatment is the same as in [2]. We choose sample points p for E and q for F with $q_x = p_x \in \mathbb{Q}$ (for vertical segments, we choose $p_y = q_y \in \mathbb{Q}$), and consider the planar curve $f|_{x=p_x} := f(p_x, y, z) \in \mathbb{Q}[y, z]$. The i-th lift $F^{(i)}$ of F is adjacent to the j-th lift $E^{(j)}$ of E if and only if there is an arc of the curve $V(f|_{x=p_x})$ connecting the i-th point over q_y with the j-th point over p_y. In our implementation, we use the algorithm presented in [7] to compute the adjacency information for $V(f|_{x=p_x})$.

C_1 has dimension 0: Then, C_1 is a vertex at point p, and C_2 is either an edge or a face. As above, we can filter the case that f_p has at most one multiple real root.

For the general method, let $z_1, \ldots, z_m$ denote the real roots of f_p. We choose (rational) intermediate values $q_0, \ldots, q_m$ such that $q_{i-1} < z_i < q_i$ for all $i = 1, \ldots, m$. The planes $z = q_i$ divide the real space in $m + 2$ *buckets* that separate the stack points z_i.

Definition 4 *Let $C \in \mathcal{A}_S$ (edge or face) be adjacent to p. A point p' on C is* bucket-faithful *if there exists a path from p' to p on C such that on that path, each lift $C^{(i)} \in \Omega_S$ over C remains in the same bucket.*

With a bucket-faithful point p' on C, the adjacencies of cells over C with cells over p follows by considering the real roots of $f_{p'}$: if the i-th root of $f_{p'}$ lies in the bucket of z_j, then the cells $C^{(i)}$ and $p^{(j)}$ are adjacent. Furthermore, points over p' that lie in either the bottom- or the top-most bucket belong to asymptotic components, i.e., they are unbounded in z-direction.

To compute a bucket-faithful point for C, we first compute a box B containing p such that no intersection with any plane $z = q_i$ takes place over B. In other words, each continuous path on S over B remains in the same bucket. We shrink B further until all features of $\mathcal{A}_S$ adjacent to p intersect the boundary of B. To find a bucket-faithful point of an edge adjacent to p, we start at p, and follow the edge until it crosses B for the first time. This intersection point is bucket faithful. For a bucket-faithful point of a face F, consider the edge $E \in \mathcal{A}_S$ that precedes F in counterclockwise order around p. Let q_E be the first intersection of E with the box boundary. Choose a point q_F on the box boundary between q_E and the next intersection of the box boundary with Γ_S in clockwise order. q_F is bucket-faithful for F.

Vertical lines In case where S contains a vertical line ℓ_p at a point $p \in \mathbb{R}^2$, the lift $p^{(i)}$, and thus the cell decomposition Ω_S as defined in Definition 3, is

Instance	$\deg_{x,y,z}$	(#V,#E,#F)	$\lvert\Omega_S\rvert$	**t** (in s)
steiner-roman	2,2,2	(5,12,8)	28	**0.73**
cayley-cubic	2,2,2	(3,10,8)	31	**0.74**
tangle-cube	4,4,4	(0,6,7)	28	**0.61**
bohemian-dome	4,4,4	(7,20,14)	61	**0.75**
chair	4,4,4	(4,9,7)	31	**3.05**
hunt	6,6,6	(3,2,3)	15	**1.21**
spiky	6,9,6	(1,8,8)	13	**1.43**
C8	8,8,8	(40,48,26)	496	**30.95**
random-3	3,3,3	(2,3,3)	15	**0.17**
random-4	4,4,4	(7,14,8)	64	**4.50**
random-5	5,5,5	(16,24,10)	154	**236.40**
interpolated-3	3,3,3	(4,6,3)	23	**0.34**
interpolated-4	4,4,4	(12,18,9)	82	**31.41**
projection-4d	4,4,4	(4,12,9)	34	**10.33**

no longer well-defined. At such points, ℓ_p is added to the cell decomposition. However, in order to fulfill the boundary property, ℓ_p is decomposed into vertical segments, and separating points in-between, according to the following theorem.

Theorem 6 *Let S contain the vertical line ℓ_p and $F \in \mathcal{A}_S$ be a face, which is adjacent to p. Then for any surface patch $F^{(i)}$ (the j-th lift of F) there exists an interval $I(F^{(i)}) \subset \mathbb{R}$, such that $p \times I(F^{(i)}) = \overline{F^{(i)}} \cap \ell_p$.*

The separating points are given by algebraic equations. The adjacency for cells over p is computed similarly to the case of non-vertical vertices, as described above. Because of space limitations, details are omitted here but are discussed in [3].

5 Implementation, conclusions, and outlook

We implemented the analysis in C++, taking from Exacus the surface representation and the analyses of algebraic curves [6, 7] and combined them with Cgal's `Arrangement_2` package to construct the (n,k)-arrangement. The possibility to attach data to Dcel-features allows to efficiently access (n,k)-relevant data for the lifting step. All computations follow the lazy-evaluation scheme, i.e., they are only triggered on demand and cached, e.g., the lifting. Following the generic programming paradigm, we decoupled combinatorial tasks from surface-specific ones.

We run experiments on well-known examples from algebraic geometry, interpolated instances, and also a generic projection of two quadrics in 4D; executed on a AMD Opteron(tm) 8218 (1 GHz) multiprocessor (1 MB cache) platform (32 GB RAM) running Debian Etch, compiled with `g++-4.1.2` using flags `-O2 -DNDEBUG` and the exact number types of Core. The table presents example surfaces along with their structural data and the obtained running times. About 90% of the time is spent to construct $\mathcal{A}_S$. Some surfaces do not show any (n,k)-vertex (e.g., `tangle-cube`) or -edge (e.g., xy-functional surfaces) at all. Due to our approximative and combinatorial methods, not more than the remaining 10% are spent to compute liftings and adjacencies.

Our work demonstrates that surface analysis is practically feasible for moderate degrees without switching to a generic position. The experiments show promising results thanks to our minimalistic cell decomposition and the consequent application of approximate methods. We are currently investigating how to enhance the cell decomposition to produce exact triangulations of arbitrary surfaces. An extension to multiple surfaces enables to analyze space curves and to realize boolean operations for surfaces.

References

[1] D. S. Arnon, G. E. Collins, and S. McCallum. Cylindrical algebraic decomposition I: The basic algorithm. *SIAM Journal on Computing*, 13:865–877, 1984.

[2] D. S. Arnon, G. E. Collins, and S. McCallum. An adjacency algorithm for cylindrical algebraic decompositions of three-dimensional space. *Journal of Symbolic Computation*, 5:163–187, 1988.

[3] E. Berberich, M. Kerber, M. Sagraloff. Exact Geometric-Topological Analysis of Algebraic Surfaces. In *Proc. of the 24th Ann. Symp. on Computational Geometry (SCG'08)*, 2008. To appear.

[4] J.-S. Cheng, X.-S. Gao, and M. Li. Determining the topology of real algebraic surfaces. In R. Martin, H. Bez, and M. Sabin, editors, *11. IMA Conference on the Mathematics of Surfaces*, volume 3604 of *LNCS*, pages 121–146, 2005.

[5] D. Diochnos, I. Z. Emiris, and E. P. Tsigaridas. On the complexity of real solving bivariate systems. In C. W. Brown, editor, *Proc. of the 2007 International Symp. on Symbolic and Algebraic Computation (ISSAC 2007)*, pages 127–134, 2007.

[6] A. Eigenwillig and M. Kerber. Exact and efficient 2d-arrangements of arbitrary algebraic curves. In *Proc. of the Nineteenth Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA08)*, pages 122–131, 2008.

[7] A. Eigenwillig, M. Kerber, and N. Wolpert. Fast and exact geometric analysis of real algebraic plane curves. In C. W. Brown, editor, *Proc. of the 2007 International Symp. on Symbolic and Algebraic Computation (ISSAC 2007)*, pages 151–158, 2007.

[8] A. Eigenwillig, L. Kettner, W. Krandick, K. Mehlhorn, S. Schmitt, and N. Wolpert. A Descartes algorithm for polynomials with bit-stream coefficients. In *8th International Workshop on Computer Algebra in Scientific Computing (CASC 2005)*, volume 3718 of *LNCS*, pages 138–149, 2005.

[9] L. Gonzalez-Vega, T. Recio, H. Lombardi, and M.-F. Roy. Sturm-Habicht sequences, determinants and real roots of univariate polynomials. In B. Caviness and J. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, Texts and Monographs in Symbolic Computation, pages 300–316. Springer, 1998.

[10] B. Mourrain and J.-P. Técourt. Isotopic meshing of a real algebraic surface. Technical Report 5508, INRIA Sophia-Antipolis, 2005.

The Challenge of 3D Photo/Cinematography to Computational Geometry

Jean Ponce

WILLOW project-team
DI - École Normale Supérieure
45, rue d'Ulm
75230 Paris cedex 05, France

e-mail: Jean.Ponce@ens.fr
URL: http://www.di.ens.fr/~ponce/

Abstract

I will present in this talk some recent results on the automated acquisition of 3D object and scene models from multiple photographs (a process known as 3D photography) and the recovery of both deformable shapes and dense velocity fields from video sequences (3D cinematography). I will also discuss a number of challenges and issues raised by this work that may be of interest to computational geometry researchers, for example: What is an effective formulation for the surface reconstruction problem with visibility constraints? How can internal contour information be exploited in "visual-hull-like" models? Are there effective representations of aspect graphs of polyhedral objects?

This is joint work with Yasutaka Furukawa.

Improved Upper Bounds on the Number of Vertices of Weight $\leq k$ in Particular Arrangements of Pseudocircles

Ronald Ortner*

Abstract

In arrangements of pseudocircles (Jordan curves) the weight of a vertex (intersection point) is the number of pseudocircles that contain the vertex in its interior. We give improved upper bounds on the number of vertices of weight $\leq k$ in certain arrangements of pseudocircles in the plane.

1 Introduction

A *pseudocircle* is a simple closed (Jordan) curve in the plane. An *arrangement of pseudocircles* is a finite set $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ of simple closed curves in the plane such that (i) no three curves meet each other at the same point, (ii) each two curves γ_i, γ_j have at most two points in common, and (iii) these *intersection points* in $\gamma_i \cap \gamma_j$ are always points where γ_i, γ_j cross each other. An arrangement is *complete* if each two pseudocircles intersect.

Any arrangement can be interpreted as a planar embedding of a graph whose vertices are the intersection points between the pseudocircles and whose edges are the curves between these intersections. In the following we will often refer to this graph when talking about *vertices*, *edges*, and *faces* of the arrangement.

Definition 1 *Let $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ be an arrangement of pseudocircles. The* weight of a vertex V *is the number of pseudocircles γ_i such that V is contained in* $\mathrm{int}(\gamma_i)$*, the interior of γ_i. Weights of edges and faces are defined accordingly.*

We will consider the number $v_k = v_k(\Gamma)$ of vertices of given weight k, the number $v_{\leq k} = v_{\leq k}(\Gamma)$ of vertices of weight $\leq k$, and the number $v_{\geq k} = v_{\geq k}(\Gamma)$ of vertices of weight $\geq k$. Further, $f_k = f_k(\Gamma)$ denotes the number of faces of weight k.

Concerning the characterization of the *weight vectors* $(v_0, v_1, \ldots, v_{n-2})$ of arrangements of pseudocircles little is known. So far, sharp upper bounds on v_k exist only for $k = 0$.

Theorem 1 (Kedem et al. [2]) *For all arrangements Γ with $n := |\Gamma| \geq 3$,*

$$v_0 \;\leq\; 6n - 12.$$

Moreover, for each $n \geq 3$ there is an arrangement of n (proper) circles in the plane such that $v_0 = 6n - 12$.

Theorem 1 can be used to obtain general upper bounds on $v_{\leq k}$ by some clever probabilistic methods.

Theorem 2 (Sharir [5]) *For all arrangements of n pseudocircles and all $k > 0$,*

$$v_{\leq k} \;\leq\; 26kn.$$

On the other hand, J. Linhart and Y. Yang established the following sharp upper bound on $v_{\geq k}$.

Theorem 3 (Linhart, Yang [4]) *For all arrangements of $n \geq 2$ pseudocircles and all k with $0 \leq k \leq n-2$,*

$$v_{\geq k} \;\leq\; (n+k)(n-k-1).$$

In this paper we are going to improve the upper bounds of Theorems 1 and 2 for some particular classes of arrangements.

2 Preparations and first results

2.1 Improved bounds from Theorem 3

We start with a bound due to J. Linhart, which holds if there is a face of large weight in the arrangement. It is based upon the following result of Y. Yang (which can be shown by turning the arrangement in question inside out).

Proposition 4 (Yang [6]) *Let Γ be an arrangement of n pseudocircles in the plane with weight vector $(v_0, v_1, \ldots, v_{n-2})$ and $f_n > 0$. Then there is an arrangement of pseudocircles Γ' with weight vector $(v'_0, v'_1, \ldots, v'_{n-2}) = (v_{n-2}, v_{n-1}, \ldots, v_0)$.*

J. Linhart [3] pointed out that Proposition 4 together with Theorem 3 yields the following improvement of the upper bound on $v_{\leq k}$ for arrangements with $f_n > 0$.

Theorem 5 (Linhart [3]) *For all arrangements Γ of n pseudocircles with $f_n > 0$,*

$$v_{\leq k} \;\leq\; 2(k+1)n - (k+1)(k+2).$$

*Department Mathematik und Informationstechnologie, Montanuniversität Leoben, rortner@unileoben.ac.at

Proof. Let Γ be an arrangement with weight vector $(v_0, v_1, \ldots, v_{n-2})$ and $f_n > 0$. Then by Proposition 4, there exists an arrangement Γ' with $v'_k = v_{n-k-2}$ vertices of weight k for $0 \le k \le n-2$. Therefore, by Theorem 3,

$$\begin{aligned} v_{\le k} &= \sum_{j=0}^{k} v_j = \sum_{j=0}^{k} v'_{n-j-2} = \sum_{j=n-k-2}^{n-2} v'_j \\ &= v'_{\ge n-k-2} \\ &\le (n+n-k-2)\big(n-(n-k-2)-1\big) = \\ &= 2(k+1)n-(k+1)(k+2). \end{aligned}$$ □

Theorem 5 may be used to obtain bounds of

$$v_{\le k} \le 2n(n-w+k+1)-(k+1)(k+2),$$

if there is a face of large weight w. Of course, in general this bound is worse than that of Theorem 2 as it is quadratic in n.

2.2 Improved bounds from Theorem 1

2.2.1 Faces with many participating pseudocircles

Bounds on v_0 can also be improved if there is a face of weight 0 with many pseudocircles participating in its boundary.

Proposition 6 *Let Γ be an arrangement of n pseudocircles with a face F of weight 0 such that for each $\gamma \in \Gamma$ there is an edge of γ on ∂F, the boundary of F. Then*

$$v_0 \le 4n-6.$$

Proof. Let us assume that there exists an arrangement Γ as described in the proposition such that $v_0 > 4n-6$. As shown in Fig. 1 we can add a pseudocircle γ' to Γ such that γ' cuts each $\gamma \in \Gamma$ on ∂F in two vertices of weight 0. Note that we may add γ' such that it does not contain any vertices of Γ in its interior. Hence, in the arrangement $\Gamma' := \Gamma \cup \{\gamma'\}$ we have

$$v_0(\Gamma') > 4n-6+2n = 6(n+1)-12,$$

which contradicts Theorem 1. □

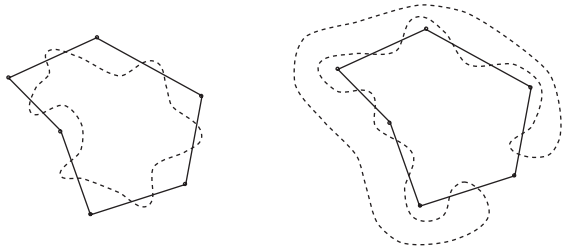


Figure 1: Adding a pseudocircle γ' cutting each $\gamma \in \Gamma$ in two vertices of weight 0.

This proof method can be generalized to obtain a bound of $v_0 \le 6n-2m-6$ for arrangements of n pseudocircles with a face of weight 0 in whose boundary m pseudocircles participate.

As shown in Fig. 2, the bound of Proposition 6 is sharp.

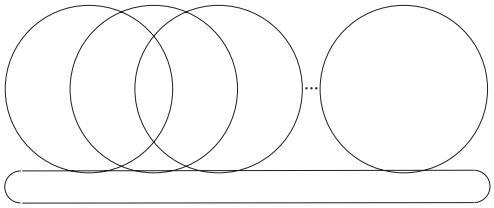


Figure 2: Arrangement of n pseudocircles with $v_0 = 4n-6$. Note that each pseudocircle participates in the unbounded face of weight 0.

2.2.2 A bound depending on f_0

Theorem 1 can also be used to obtain an upper bound on v_0 that depends on the number f_0 of faces of weight 0.

Theorem 7 *Let Γ be an arrangement of n pseudocircles. Then*

$$v_0 \le 2n+2f_0-4.$$

Theorem 7 can be proved from Theorem 1 with the aid of the upper bound on f_0 given in Proposition 8 below, which is also a direct consequence of Theorem 1. As Theorem 7 together with Proposition 8 entails Theorem 1, this can be considered as self-strengthening of Theorem 1.

Proposition 8 *Let Γ be an arrangement of $n \ge 3$ pseudocircles. Then*

$$f_0 \le 2n-4.$$

Proof. First note that the boundary of each face of weight 0 consists of at least three edges (and hence vertices) of weight 0. For if there were a face with only two edges belonging to some pseudocircles γ_i and γ_j, then $\gamma_i \cap \gamma_j$ would have more than the two allowed intersection points. On the other hand, each vertex of weight 0 is on the boundary of only a single face of weight 0. Therefore by Theorem 1,

$$f_0 \le \frac{v_0}{3} \le \frac{6n-12}{3} = 2n-4.$$ □

Proof of Theorem 7. If f_0 equals the maximal value $2n-4$ of Proposition 8, the theorem holds by Theorem 1. We proceed by induction on f_0, assuming that the theorem holds for all arrangements with

$f_0 + 1$ faces of weight 0. Given an arbitrary arrangement with f_0 faces of weight 0, one can easily add a pseudocircle γ that separates a face of weight 0 into two faces of weight 0 without covering any vertices of Γ (cf. also the construction in the proof of Proposition 6). The arising arrangement $\Gamma' := \Gamma \cup \{\gamma\}$ consists of $n + 1$ pseudocircles and has $f_0 + 1$ faces of weight 0. Applying the induction assumption, we have

$$\begin{aligned} v_0(\Gamma) &= v_0(\Gamma') - 4 \leq 2(n+1) + 2(f_0+1) - 8 \\ &= 2n + 2f_0 - 4. \end{aligned}$$ □

3 Improved bounds for complete arrangements with forbidden subarrangements

In this section we consider bounds for complete arrangements. First, we'd like to remark that the bound of Theorem 1 is sharp for complete arrangements, too. That is, for each $n \geq 3$ there is a complete arrangement with $v_0 = 6n - 12$ (see Fig. 3).

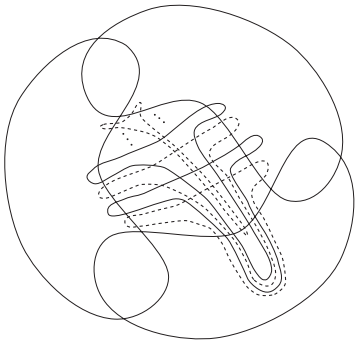


Figure 3: A complete arrangement of pseudocircles with $v_0 = 6n - 12$.

Thus, in order to obtain improved bounds on v_0, one has to put some additional restrictions on the arrangement, e.g. by forbidding certain subarrangements, which will be considered in the following.

3.1 Forbidding α-subarrangements

Evidently, arrangements of three pseudocircles are the smallest subarrangements of interest in this respect. Figure 4 shows the four different types one has to take into account.

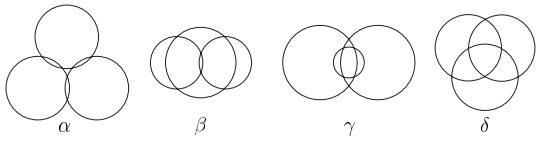


Figure 4: Complete arrangements of three pseudocircles in the plane.

Subarrangements of type α play a special role here. Not only are they the only arrangements of three pseudocircles which meet the bound of Theorem 1. They are also the only complete arrangements of three pseudocircles without any face of weight 3, which is of importance in the light of the following Helly type theorem.

Theorem 9 (Helly [1]; Kerékjártó) *Let $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ be an arrangement of pseudocircles such that for all pairwise distinct $\gamma_i, \gamma_j, \gamma_k$,*

$$\operatorname{int}(\gamma_i) \cap \operatorname{int}(\gamma_j) \cap \operatorname{int}(\gamma_k) \neq \varnothing.$$

Then

$$\bigcap_{i=1}^{n} \operatorname{int}(\gamma_i) \neq \varnothing.$$

Corollary 10 *Let Γ be a complete arrangement of $n \geq 2$ pseudocircles that has no subarrangement of type α. Then*

$$v_{\leq k} \leq 2(k+1)n - (k+1)(k+2).$$

Proof. Since Γ has no α-subarrangement, the condition in Theorem 9 holds, and we may conclude that there is a face of weight n in Γ. Applying Theorem 5 yields the claimed bound. □

3.2 Forbidding α^4-subarrangements

It is a natural question whether there are alternative bounds for other forbidden subarrangements as well. The unique complete arrangement of four pseudocircles that meets the bound of Theorem 1 seems to be a good candidate. In such an *α^4-arrangement* each subarrangement of three pseudocircles is of type α. α^4-arrangements prominently appear in the arrangement of Fig. 3, where the three outer pseudocircles together with any other pseudocircle form an α^4-arrangement. Indeed, for α^4-free arrangements in which there is also no β-subarrangement (cf. Fig. 4) we can show the following improved upper bound on v_0.

Theorem 11 *In complete arrangements of $n \geq 2$ pseudocircles that are α^4-free and β-free,*

$$v_0 \leq 4n - 6.$$

Theorem 11 follows immediately from the following bound on f_0 together with Theorem 7.

Theorem 12 *In complete arrangements of $n \geq 2$ pseudocircles that are α^4-free and β-free,*

$$f_0 \leq n - 1.$$

For the proof of Theorem 12 the following lemma is useful. We skip a proof.

Lemma 13 *Let Γ be a complete, β-free arrangement. Then for each face F of weight 0 in Γ there is a unique α-arrangement $\Gamma_\alpha \subseteq \Gamma$ such that F is the bounded face of weight 0 in Γ_α. In particular, each face of weight 0 has only three edges.*

Proof of Theorem 12. We give a proof by induction on $n := |\Gamma|$. The case $n = 2$ is trivial, while for $n = 3$ one may consult Fig. 4. If $n > 3$, choose an arbitrary pseudocircle γ in Γ. By induction assumption the theorem holds for $\Gamma' := \Gamma \setminus \{\gamma\}$. We claim that adding γ to Γ' will increase f_0 by at most 1. Indeed, f_0 could be increased by more than 1 only in one of the following two cases:

First, γ may separate a single face F of weight 0 in Γ' into more than two new faces of weight 0. By Lemma 13 such a face F has only three edges which belong to three pseudocircles that form an α-arrangement Γ_α. Thus, in order to separate F as described above, γ has to intersect each pseudocircle of Γ_α in two vertices of weight 0, so that $\Gamma_\alpha \cup \{\gamma\}$ would be a forbidden α^4-arrangement.

On the other hand, there might be two distinct faces F_1, F_2 in Γ', such that γ separates each F_i into two new faces of weight 0. By Lemma 13, there is a unique α-arrangement Γ_α enclosing F_1, so that F_2 will be outside Γ_α (i.e. contained in the unbounded face of weight 0 of Γ_α). Hence, γ would have to intersect the bounded as well as the unbounded face of weight 0 in Γ_α. But it is easy to see that this can only happen if γ together with two pseudocircles in Γ_α forms a forbidden β-subarrangement. □

The bounds of Theorems 11 and 12 are sharp. Take $(n-1)$ pseudocircles such that any subarrangement of three pseudocircles is of type δ. In this arrangement $f_0 = 1$, and each pseudocircle has an edge (and hence two vertices) on the single face of weight 0. Adding another pseudocircle just as indicated in the proof of Proposition 6 (cf. Fig. 1) yields an arrangement with $f_0 = n - 1$ and $v_0 = 4n - 6$.

The improved upper bound on v_0 of Theorem 11 can in turn be used to improve the upper bound on $v_{\leq k}$ for complete, α^4-free arrangements.

Theorem 14 *For complete, α^4-free and β-free arrangements of $n \geq 2$ pseudocircles and $k > 0$,*

$$v_{\leq k} \;\leq\; 18kn.$$

Proof. The proof is basically identical to the proof of Theorem 2 in [5], only with the application of Theorem 1 replaced by an application of Theorem 11 and the constants adapted accordingly. □

The bounds of Theorems 11 and 14 can easily be generalized to (not necessarily complete) β-free arrangements that do not contain certain subarrangements that are generalizations of α^4-arrangements.

4 Conclusion

We conjecture that Theorems 11, 12, and 14 also hold if we drop the condition that the arrangement is β-free, i.e., for the improved bounds to hold it is sufficient that a complete arrangement is α^4-free. However, the topology of these arrangements quickly becomes rather involved so that we haven't yet succeeded in proving this. As an α^4-arrangement cannot be realized with unit circles, a proof of our conjecture would also imply that Theorems 11 and 14 hold in particular for complete arrangements of unit circles. As shown in Fig. 5, in this case the improved bound on v_0 would also be sharp.

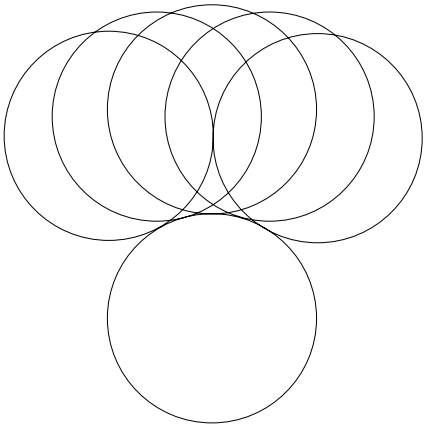


Figure 5: Complete arrangement of six unit circles with $v_0 = 4n - 6$. Points that look like touching points should be two intersection points between the respective circles. Further circles can easily be added to meet the bound for arbitrary n.

References

[1] E. Helly, Über Systeme von abgeschlossenen Mengen mit gemeinschaftlichen Punkten. *Monatsh. Math.* 37 (1930), 281–302.

[2] K. Kedem, R. Livne, J. Pach, and M. Sharir, On the Union of Jordan Regions and Collision-Free Translational Motion Amidst Polygonal Obstacles. *Discrete Comput. Geom.* 1 (1986), 59–71.

[3] J. Linhart, private communication.

[4] J. Linhart, Y. Yang, Arrangements of Arcs and Pseudocircles, *Beiträge Algebra Geom.* 37/2 (1996), 391–398 .

[5] M. Sharir, On k-Sets in Arrangements of Curves and Surfaces, *Discrete Comput. Geom.* 6 (1991), 593–613.

[6] Y. Yang, Arrangements of Circles on E^2, unpublished typescript.

Helly-Type Theorems for Approximate Covering

Julien Demouth* Olivier Devillers† Marc Glisse‡ Xavier Goaoc*

Abstract

Let $\mathcal{F} \cup \{U\}$ be a collection of convex sets in $\mathbb{R}^d$ such that $\mathcal{F}$ covers U. We show that if the elements of $\mathcal{F}$ and U have comparable size, in the sense that each contains a ball of radius r and is contained in a ball of radius R for some fixed r and R, then for any $\epsilon > 0$ there exists $\mathcal{H}_\epsilon \subset \mathcal{F}$, whose size $|\mathcal{H}_\epsilon|$ is polynomial in $1/\epsilon$ and independent of $|\mathcal{F}|$, that covers U except for a volume of at most ϵ. The size of the smallest such subset depends on the geometry of the elements of $\mathcal{F}$; specifically, we prove that it is $O(\frac{1}{\epsilon})$ when $\mathcal{F}$ consists of axis-parallel unit squares in the plane and $\widetilde{O}(\epsilon^{\frac{1-d}{2}})$ when $\mathcal{F}$ consists of unit balls in $\mathbb{R}^d$ (recall that $\widetilde{O}(n)$ means $O(n \log^\beta n)$ for some constant β), and that these bounds are, in the worst-case, tight up to the logarithmic factor .

We extend these results to surface-to-surface visibility in 3 dimensions: if a collection $\mathcal{F}$ of disjoint unit balls occludes visibility between two balls then a subset of $\mathcal{F}$ of size $\widetilde{O}(\epsilon^{-\frac{7}{2}})$ blocks visibility along all but a set of lines of measure ϵ.

Finally, for each of the above situations we give an algorithm that takes $\mathcal{F}$ and U as input and outputs in time $O(|\mathcal{F}| * |\mathcal{H}_\epsilon|)$ either a point in U not covered by $\mathcal{F}$ or a subset $\mathcal{H}_\epsilon$ covering U up to a measure ϵ, with $|\mathcal{H}_\epsilon|$ satisfying the previous bound.

1 Introduction

A family $\mathcal{F}$ of sets covers a set U if the union of the elements of $\mathcal{F}$ contains U. The classical SetCover problem asks, given a covering $\mathcal{F}$ of a finite set U, for the smallest subset of $\mathcal{F}$ that covers U. In the geometric setting, both U and the elements of $\mathcal{F}$ are subsets of a geometric space, for example points, hyperplanes or balls in $\mathbb{R}^d$. The original problem is **NP**-hard [8] and so are many of its geometric analogues. Therefore, approximation algorithms have been largely investigated, and in general, one looks for a subset of $\mathcal{F}$ that *completely* covers U and whose size is near-optimal; approximation factors better than $\log |U|$ are provably difficult to achieve in the finite case [7, 9] and constant factor approximations were obtained for only a few geometric versions [4] (see also [3]). In this paper, we relax the problem in a different direction: given a covering $\mathcal{F}$ of a set U, we look for a small subset of $\mathcal{F}$ that covers *most of* U. Specifically, in the geometric setting we define an ϵ-covering of U as a collection $\mathcal{H}$ of sets whose union covers U except for a volume of at most ϵ. Although this is a natural question, we are not aware of previous results in this direction.

Results. Let $\mathcal{F}$ be a covering of a convex set U by convex sets in $\mathbb{R}^d$. Let $\mathcal{H}_\epsilon$ denote a smallest ϵ-covering of U contained in $\mathcal{F}$. Recall that $\widetilde{O}(n)$ means $O(n \log^\beta n)$ for some β. Our main results are the following:

- If the elements in $\mathcal{F}$ have similar size, i.e. each can be sandwiched between two spheres of fixed radii, then $|\mathcal{H}_\epsilon|$ is bounded polynomially in $1/\epsilon$ and independently of $|\mathcal{F}|$ (Theorem 3).
- $|\mathcal{H}_\epsilon|$ is $O\left(\frac{1}{\epsilon}\right)$ when $\mathcal{F}$ consists of axis-parallel unit squares in the plane (Theorem 4) and $\widetilde{O}(\epsilon^{\frac{1-d}{2}})$ if $\mathcal{F}$ consists of unit balls in $\mathbb{R}^d$ (Theorem 5) or smooth convex sets of bounded curvature (Corollary 7). These bounds are tight in the worst-case (up to the logarithmic factor).
- These results extend to visibility occlusion among disjoint unit balls in $\mathbb{R}^3$, where the notion of volume used relates to the form factor (Theorem 8).
- For covering by squares or balls and visibility in 3D, we give algorithms that take $\mathcal{F}$ and U as input and output in $O(|\mathcal{F}| * |\mathcal{H}_\epsilon|)$-time either a point in U not covered by $\mathcal{F}$ or an ϵ-cover of U contained in $\mathcal{F}$; $|\mathcal{H}_\epsilon|$ denotes our bound on the size of the smallest ϵ-covering for that situation (Section 6).

Our results imply that there do not exist arbitrarily large minimal ϵ-cover of a convex set by similar-sized convex sets, which is in sharp contrast with exact covering. The order $\sqrt{\epsilon}$ gap between our bounds in the case of squares and smooth convex sets with bounded curvature in the plane shows that the asymptotic behavior of $|\mathcal{H}_\epsilon|$ when $\epsilon \to 0$ depends not only on the size but also on the shape of the covering objects.

Geometric problems such as guarding or visibility can be rephrased as covering problems where, given a

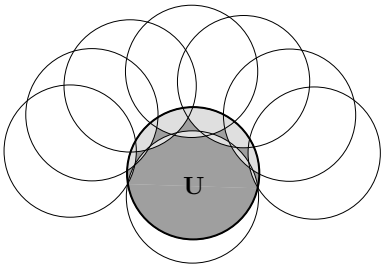
*LORIA - INRIA Grand Est, Univ. Nancy 2, Projet VEGAS, France. {demouth,goaoc}@loria.fr.

†INRIA Sophia-Antipolis, France. olivier.devillers@sophia.inria.fr.

‡Gipsa-Lab, CNRS UMR 5216, Grenoble, France. marc.glisse@normalesup.org

collection $\mathcal{F}$ and a set U one has to decide if $\mathcal{F}$ covers U. Such tests can be expensive, e.g. no algorithm with complexity $o(n^4)$ is known for reporting visible pairs among n triangles in $\mathbb{R}^3$ [10, Problem 7.7.1(f)], so approximation algorithms are often used in practice. Our algorithms are interesting in that they are simple, have complexity linear in $|\mathcal{F}|$ and allow to control the error a priori.

Helly-type theorems. Helly's theorem asserts that n convex sets in $\mathbb{R}^d$ have non-empty intersection if any $d+1$ of them have non-empty intersection. Results of similar flavor – that some property on a set $\mathcal{F}$ can be checked by examining its subsets of bounded size – are known as *Helly-type theorems* and are the object of active research [5, 6, 14]. A collection $\mathcal{F}$ covers U



if and only if the intersection of the complement of its elements and U is empty; thus, if $\mathcal{F}$ consists of complement of convex sets in $\mathbb{R}^d$ and covers a convex set U, then $d+1$ elements in $\mathcal{F}$ suffice to cover U. Cases where such statements are known are, however, rather exceptional as for most classes of objects there exists arbitrarily large minimal covering families (the figure above illustrates the principle of such a construction for unit disks). Our Theorems 3, 4, 5 and 8 show that the situation is different when *approximate* covering is considered.

Due to the lack of space this article does not contain all the proofs. They are available in the full version [11].

2 The general case

We start with a simple observation on approximation of a convex set by a grid.

Lemma 1 *Let $\mathcal{O} \subset \mathbb{R}^d$ be a convex set of diameter at most R and Γ a regular grid of step ℓ. The cells of Γ contained in $\mathcal{O}$ cover $\mathcal{O}$ except for a volume of $O(\ell)$.*

Note that the constant hidden in the $O()$ notation depends on R, which is fixed for a given collection of sets.

A collection $\mathcal{F}$ of sets has *scale* (r, R) if each element in $\mathcal{F}$ contains a ball of radius r and is contained in one of radius R. We define $\kappa = r/(16R\sqrt{d})$ and prove the following technical lemma:

Lemma 2 *If U is a cube of side length ℓ in $\mathbb{R}^d$ and $\mathcal{O}$ is a convex set of scale (r, R), such that $\ell \leq 2r$, containing the center of U, then $\mathcal{O} \cap U$ contains at least one cell of any regular grid of step at most $\kappa\ell$.*

We can now state the main result of this section:

Theorem 3 *For any d, r and R, there exists a polynomial function $H(\epsilon) = H_{d,r,R}(\epsilon)$ such that the following holds. Any covering $\mathcal{F}$ of a convex set $U \subset \mathbb{R}^d$ of diameter at most R by a collection of convex sets of scale (r, R) contains an ϵ-covering of U of size at most $H(\epsilon)$.*

Proof. Let $\mathcal{R}_0$ be an $\frac{\epsilon}{2}$-covering of U by $O(\epsilon^{-d})$ cells of a regular grid; Lemma 1 guarantees its existence. We then proceed recursively. At step i, we have a subset $\mathcal{C}_i$ of $\mathcal{F}$ and a set $\mathcal{R}_i$ of congruent cubes, each of side length $\ell_i = \kappa^i \ell_0$, that together form an $\epsilon/2$-cover of U. For each cube $Y \in \mathcal{R}_i$, we select an object in $\mathcal{F}$ that covers its center and add it to $\mathcal{C}_{i+1}$; we then subdivide Y using a grid of step $\kappa\ell_i$ and collect the cubes not covered by C_{i+1} into $\mathcal{R}_{i+1}$. We initialize the recursion with $\mathcal{R}_0$ and $\mathcal{C}_0 = \emptyset$. Lemma 2 implies that in the subdivision of any cube, at least one of the smaller cubes is covered, and thus

$$|\mathcal{R}_{i+1}| \leq |\mathcal{R}_i|(\kappa^{-d} - 1) \quad \text{and} \quad |\mathcal{C}_{i+1}| \leq |\mathcal{C}_i| + |\mathcal{R}_i|.$$

After some computations we get that $|C_i| = O\left(\epsilon^{-O(d^2\kappa^{-d}\log\frac{1}{\kappa})}\right)$, which concludes the proof. □

This result is optimal in the sense that it becomes false if one of the scale or convexity conditions is dropped. While a more careful analysis might improve the bound obtained, and in particular the dependency of the exponent of $1/\epsilon$ on d, the next sections show that pinning down the precise asymptotic behavior of $H(\epsilon)$ requires taking into account the shape of the objects in $\mathcal{F}$.

3 Covering by squares

For axis parallel boxes in $\mathbb{R}^d$, the analysis of the previous section holds for $\kappa = 1/2$; if, moreover, U is a cube, then $|\mathcal{R}_0|$ is 1 and this bound becomes $O\left(\epsilon^{-O(d2^d)}\right)$. We improve this bound in the planar case:

Theorem 4 *Let $U \subset \mathbb{R}^2$ be an axis-parallel square of side r covered by a finite collection $\mathcal{F}$ of larger axis-aligned squares. For $\epsilon > 0$ sufficiently small, the smallest ϵ-covering of U contained in $\mathcal{F}$ has size $O\left(\frac{1}{\epsilon}\right)$; this bound is tight in the worst-case.*

4 Covering by balls

When the objects of $\mathcal{F}$ are balls in $\mathbb{R}^d$, we can prove the following, almost tight, bound:

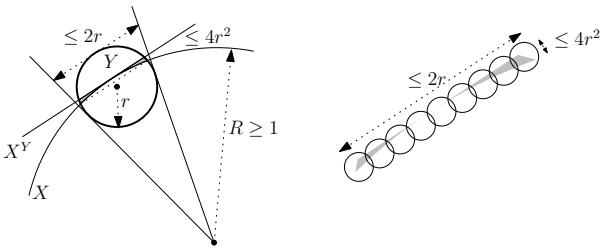
Theorem 5 *Let $\mathcal{F}$ be a covering of a convex $U \subset \mathbb{R}^d$ of diameter at most R by finitely many balls, each of radius at least r. For any $\epsilon > 0$, the smallest ϵ-covering of U contained in $\mathcal{F}$ has size $\widetilde{O}\left(\epsilon^{\frac{1-d}{2}}\right)$. This bound is tight up to the logarithmic factor in the worst-case.*

The proof of this theorem relies on a technical lemma, presented here in the planar case ($d = 2$). The general case is similar.

For two disks X and Y, we denote by X^Y the half-plane containing X and bounded by the tangent to X at the projection[1] of the center of Y on the boundary of X. We denote by $\mathcal{F}^Y$ the collection $\{X^Y \mid X \in \mathcal{F}\}$.

Lemma 6 *Let Y be a disk of radius $r < 1$ and $\mathcal{F}$ a covering of a unit disk U by larger disks. Then, $U \cap Y$ can be covered by a triple $C(Y) \subset \mathcal{F}$ and a collection $R(Y)$, of at most $\frac{3}{r}$ disks of radius $4r^2$.*

Proof. Since the collection $\mathcal{F}^Y$ covers U, it also covers $U \cap Y$ and, by Helly's theorem, three of these half-planes must cover $U \cap Y$. We denote by $C(Y)$ the corresponding disks in $\mathcal{F}$. For any disk $X \in \mathcal{F}$, the area $\left(X^Y \cap Y\right) \setminus (X \cap Y)$ is inscribed in a rectangle (figure below, on the left) with sides respectively smaller than $2r$ and $4r^2$. This rectangle can thus be



covered by overlapping disks of radius $4r^2$ centered on its larger axis (figure above, on the right). By choosing the disks so that the height covered at the intersection between two disks is, at least, $4r^2$, we need only $\frac{1}{r}$ disks. □

4.1 Smooth convex sets

The d-dimensional case of Lemma 6 requires that (i) given a ball Y, the set $U \cap Y$ be convex and that (ii) the difference between $X^Y \cap Y$ and $X \cap Y$ can be covered by $O(\frac{1}{r})$ balls of radius $O(r^2)$. If an object is convex and its boundary has a curvature of bounded norm, then for any point M on this boundary the object contains a ball (of radius bounded away from 0) and is contained in a half-space delimited by a hyperplane tangent to both the object and the ball in M; this means that covering the region between the ball and the hyperplane is enough to cover the region between the object and the hyperplane. Theorem 5 thus extends to:

Corollary 7 *Let $U \subset \mathbb{R}^d$ be a convex set of diameter at most R and $\mathcal{F}$ a covering of U by smooth convex sets whose curvatures have a norm at most γ. For any $\epsilon > 0$, the smallest subset of $\mathcal{F}$ that is an ϵ-covering of U has size $\widetilde{O}\left(\epsilon^{\frac{1-d}{2}}\right)$.*

5 Visibility among 3D unit balls

Two among n objects are *visible* if they support the endpoint of a segment that intersects no other object, and such a segment is called a *visibility segment*. Visibility between objects can be recast as a covering problem by observing that two objects are mutually visible if and only if the set of segments they support is not covered by the set of segments supported by these two objects and intersecting some other object. Yet, it is not clear whether Theorem 3 applies in this setting. In this section we show that Theorem 5 yields a similar result for visibility among balls.

A natural "volume" to quantify approximate visibility between two objects – similarly to the ϵ-coverings discussed so far – is given by the measure of the set of lines supporting visibility segments between these two objects. In fact, this corresponds, up to normalization, to the *form factor* used in computer graphics (when constant basis functions are used) to quantify visibility for simulating illumination. We call this measure the *amount of visibility* between the two objects. Building on Theorem 5, we prove:

Theorem 8 *Let $\mathcal{F} \cup \{A, B\}$ be a collection of disjoint unit balls in $\mathbb{R}^3$ such that A and B are mutually invisible. For any $\epsilon > 0$, there exists a subset $\mathcal{G}_\epsilon \subset \mathcal{F}$, of size $\widetilde{O}\left(\epsilon^{-\frac{7}{2}}\right)$, such that the amount of visibility between A and B in $\mathcal{G}_\epsilon \cup \{A, B\}$ is $O(\epsilon)$.*

6 Algorithms

The proofs of Theorems 4, 5 and 8 are constructive provided that $C(Y)$ and $R(Y)$ can be effectively computed. As in previous sections, we consider here d as a constant.

Covering by squares. In the case of covering by squares, the sets $C(Y)$ and $R(Y)$ can be computed

[1] If the two disks have the same center, we can choose any tangent to X.

trivially in $O(|\mathcal{F}|)$ time. We thus have the following consequence:

Corollary 9 *Given a covering $\mathcal{F}$ of a unit square U by unit squares, we can compute in $O\left(\frac{|\mathcal{F}|}{\epsilon}\right)$-time a point in U not covered by $\mathcal{F}$ or an ϵ-cover of U of size $O\left(\frac{1}{\epsilon}\right)$ contained in $\mathcal{F}$.*

Covering by balls. In the case of covering by balls, the main difficulty is to compute $C(Y)$, $R(Y)$ following immediately. Helly's theorem yields that given a collection $\mathcal{F}^Y$ of n halfspaces and a ball $Y \subset \mathbb{R}^d$, either there are $d+1$ halfspaces in $\mathcal{F}^Y$ that cover Y or there is a point in Y not covered by any half-space in $\mathcal{F}^Y$. In the case of covering of a ball Y by balls $\mathcal{F}$, finding $C(Y)$ reduces in $O(|\mathcal{F}|)$ time into solving the associated computational problem: finding such $d+1$ half-spaces or such a point.

Recall that *LP-type problems* are a special class of optimization problems [13]. Using a technique introduced by Amenta [1, 2], we can formulate the above problem as a LP-type problem. As a consequence, we obtain:

Corollary 10 *Let $\mathcal{F}$ be a covering of a unit ball $U \subset \mathbb{R}^d$ by unit balls. We can compute a point in U not covered by $\mathcal{F}$ or an ϵ-cover of U of size $\widetilde{O}\left(\epsilon^{\frac{1-d}{2}}\right)$ contained in $\mathcal{F}$ in time $\widetilde{O}\left(|\mathcal{F}|\epsilon^{\frac{1-d}{2}}\right)$.*

Visibility among unit balls. Corollary 10 makes the proof of Theorem 8 constructive and we get:

Corollary 11 *Let $\mathcal{F}$ be a collection of disjoint unit balls in $\mathbb{R}^3$ and let A and B be two unit balls. We can compute in $\widetilde{O}\left(|\mathcal{F}|\epsilon^{-\frac{7}{2}}\right)$-time a visibility segment between A and B or a subset $\mathcal{G}_\epsilon \subset \mathcal{F}$, of size $\widetilde{O}\left(\epsilon^{-\frac{7}{2}}\right)$, such that the amount of visibility between A and B in $\mathcal{G}_\epsilon \cup \{A, B\}$ is $O(\epsilon)$.*

7 Conclusion

We showed that the size of the smallest ϵ-covering contained in a covering $\mathcal{F}$ of a set U can be bounded polynomially in $1/\epsilon$ and independently of $|\mathcal{F}|$ when all sets are convex and the size of the sets in $\mathcal{F}$ are comparable with that of U. The order $\sqrt{\epsilon}$ gap between our bounds for smooth sets and squares indicate that the asymptotic behavior of the size of the smallest ϵ-covering depends on the shape of the objects. Do other simple shapes lead to different bounds?

These bounds yield simple and efficient algorithms for, given a family $\mathcal{F}$ and a set U, certifying either that $\mathcal{F}$ does not cover U or that $\mathcal{F}$ misses at most a volume ϵ of U. We gave an application to approximate 3D visibility, with an algorithm to decide in linear time if two balls are visible or if their form factor is at most ϵ. A natural continuation would be to compare these results to the provable bounds on the error provided by methods for approximating visibility queries used in application areas, e.g. sampling and point-to-point visibility in computer graphics.

References

[1] N. Amenta. Helly theorems and generalized linear programming. Ph.D. thesis, U.C. Berkeley, 1993.

[2] N. Amenta. Helly-type theorems and generalized linear programming. *Discrete and Computational Geometry*, 12:241–261, 1994.

[3] H. Bronnimann and M.T. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete and Computational Geometry*, 14:463–479, 1995.

[4] K.L. Clarkson and K. Varadarajan. Improved approximation algorithms for geometric set cover. *Discrete and Computational Geometry*, 37:43–58, 2007.

[5] L. Danzer, B. Grünbaum and V. Klee. Helly's theorem and its relatives. V. Klee editor, *Convexity*, Proc. of Symposia in Pure Math, 101–180, 1963.

[6] J. Eckhoff. Helly, Radon and Caratheodory type theorems. In J.E. Goodman and J. O'Rourke, editors, *Handbook of Convex Geometry*, 389–448, 1993.

[7] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[8] R. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations, Proc. Sympos. IBM Thomas J. Watson Res. Center*, 85–103, 1972.

[9] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41(5):960–981, 1994.

[10] J. Pach and M. Sharir. Combinatorial Geometry with Algorithmic Applications – The Alcala Lectures. Alcala (Spain), August 31 - September 5, 2006.

[11] J. Demouth, O. Devillers, M. Glisse and X. Goaoc. Helly-type theorems for approximate covering. Research Report n^o 6342, INRIA, Oct. 2007. Available on http://hal.inria.fr/inria-00179277/fr/.

[12] R. Seidel. Small-dimensional linear programming and convex hulls made easy. *Discrete and Computational Geometry*, 6(5):423–424, 1991.

[13] M. Sharir and E. Welzl. A combinatorial bound for linear programming and related problems. In *STACS '92: Proceedings of the 9th Annual Symposium on Theoretical Aspects of Computer Science*, 569–579, 1992.

[14] R. Wenger. Helly-type theorems and geometric transversals. In J.E. Goodman and J. O'Rourke, editors, *Handbook of Discrete & Computation Geometry*, 2nd edition, 73–96, 2004.

Dynamic Free-Space Detection for Packing Algorithms

Tobias Baumann* Magnus Jans† Elmar Schömer* Christian Schweikert* Nicola Wolpert†

Abstract

We present easy-to-implement incremental algorithms for computing the union of axis-aligned boxes. These algorithms can effectively be used for the implementation of packing algorithms which try to fit differently sized axis aligned boxes into a container modelled as a fixed point cloud.

1 Introduction

Many industries have to find arrangements of objects within a restricted amount of space. In car manufacturing, the luggage capacity of a trunk has to be determined early in the production process. The goal is to cover as much volume as possible within the given geometry of the trunk, using boxes of given sizes. To obtain good starting solutions for further optimization, e.g. using Simulated Annealing, it is useful to restrict the orientations of the boxes to be axes aligned.

In our application, the packing problem consists of an irregularly shaped container (the luggage compartment of a car). The geometry of the container is given by its CAD-data. As the geometry usually is quite difficult, we discretize the problem by approximating the shape of the container by a point cloud $\mathcal{P}$. We focus on the US standard SAE J1100 where boxes of seven different sizes have to be packed [4].

In this work we provide solutions for the following basic operations needed by a trunk packing algorithm: Check whether a given box would fit into the remaining free space created by the container geometry and boxes already inserted, insert boxes into the trunk, delete boxes from the trunk, and compute the volume of the free space. Previous work already described a trunk packing algorithm on a uniform grid with a large cell size [1]. The big cell size was inevitable because the algorithm relied on the complete representation of a conflict graph. Due to changes in the packing algorithm it is now possible to work with a finer grid. This necessitates new data structures and algorithms for the basic operations described above. In our application a grid size of 1/10 mm is sufficient. So we assume the points to have integer coordinates.

We have developed and implemented two approaches providing data structures and algorithms for performing these operations. They both work in configuration space as proposed in [2]. The suitcase S to be inserted next is represented by its reference point R_S. Accordingly, the container $\mathcal{P}$ and also each suitcase S_j, $1 \leq j \leq m$, already inserted into the trunk is enlarged by computing the corresponding Minkowski difference $\mathcal{P} \ominus \mathcal{S}$ and $S_j \ominus S$, respectively. For testing whether the new box still fits into the trunk, we have to query whether R_S lies outside each Minkowski difference and lies inside the remaining freespace. In a preprocessing step we compute the Minkowski difference $\mathcal{P} \ominus \mathcal{S}$ for of all possible boxes S to be inserted and store them in a data structure.

We have developed two different strategies to represent freespace with the following requirements in mind:

- Efficient construction of the data structure in the preprocessing step,
- fast access to the freespaces within the container,
- volume computation for branch-and-bound application,
- and efficient addition and deletion of boxes.

The first approach always maintains the boundary of the freespace as a union of rectangles during insertions and deletions. The second one is volumetric. The freespace is stored as the union of disjoint boxes. Our runtime experiments suggest that the first approach behaves better in practice.

2 The first approach - Union of rectangles

For every possible suitcase S the Minkowski difference $\mathcal{P} \ominus \mathcal{S}$ provides the possible placements of S in the configuration space. With n points in $\mathcal{P}$ this gives n axis aligned boxes $B_1, \ldots, B_n$ in 3-space. We want to compute the union $U_n = \bigcup_{i=1}^{n} B_i$ and represent its boundary ∂U_n by a set of oriented rectangles $\mathcal{R}_n$. The problem of computing volume$(U_n) \in \mathbb{R}$ is known as Klee's measure problem [3]. Yap et al. [5] developed an efficient algorithm for the d-dimenisonal problem running in $O(n^{d/2} \log n)$ time. In contrast to their solution we do not want to compute volume(U_n) alone but also an explicit representation of U_n. If we are able to find a set of rectangles $\mathcal{R}_n$ characterizing the boundary ∂U, then it is easy to compute

*Institut für Informatik, Johannes Gutenberg-Universität Mainz

†Institut für Informatik, Hochschule für Technik Stuttgart

volume(U_n) in time $O(|\mathcal{R}_n|)$ (by summing up the volumes of all signed rectangular parallelepipeds induced by the rectangles parallel to the xy-plane).

2.1 An incremental algorithm

Suppose we have computed $\mathcal{R}_{n-1}$ as the boundary ∂U_{n-1} for the first $n-1$ boxes. We now consider the last box B_n and want to update $\mathcal{R}_{n-1}$ to $\mathcal{R}_n$. We distinguish two cases:

1. $B_n \cap \partial U_{n-1} = \emptyset$**:** In this case B_n either lies completely in the interior of U_{n-1} and thus $\mathcal{R}_n = \mathcal{R}_{n-1}$, or B_n lies completely in the exterior of U_{n-1} and thus $\mathcal{R}_n = \mathcal{R}_{n-1} \cup \partial B_n$.

2. $B_n \cap \partial U_{n-1} \neq \emptyset$**:** In this case all rectangles $R \in \mathcal{R}_{n-1}$ which lie completely in the interior (or on the boundary) of B_n can be deleted. We denote this set of rectangles $\mathcal{D} = \{R \in \mathcal{R}_{n-1} | R \subseteq B_n\}$. Let $\mathcal{S} = \{R \in \mathcal{R}_{n-1} | R \cap B_n \neq \emptyset\}$ be the set of all those rectangles in $\mathcal{R}_{n-1}$, which partially lie within B_n. Each rectangle $R \in \mathcal{S}$ has to be trimmed and decomposed into (up to four) new rectangles. These new rectangles represent $R \setminus B_n$. These new rectangles resulting from the trimming process will be denoted $\mathcal{T}$. Since parts of the rectangular boundary facets of the box B_n also contribute to the boundary ∂U_n, we examine the six arrangements of line segments which arise on the six facets of B_n when intersecting them with the rectangles from R_{n-1}. Each rectangular facet F_i is decomposed in regions, which lie inside U_{n-1} and regions outside of U_{n-1} . A vertical decomposition of the regions outside of U_{n-1} yields a set $\mathcal{A}_i$ of new rectangles, which have to be added to $\mathcal{R}_{n-1}$ in order to get $\mathcal{R}_n$.

$$\mathcal{R}_n = \mathcal{R}_{n-1} \setminus (\mathcal{D} \cup \mathcal{S}) \cup \mathcal{T} \cup \bigcup_{i=1}^{6} \mathcal{A}_i$$

2.2 Algorithmic details

After this overview of our incremental algorithm for computing the union U_n of the n axis aligned boxes we want to discuss some details which are important for an efficient and robust implementation of this algorithm. The assumption of integer coordinates for all boxes avoids robustness problems due to floating point arithmetic. All steps of the algorithm above can be performed with integer arithmetic, such that all decisions can be safely made. The next question we want to address is that of a suitable data structure for the set of rectangles $\mathcal{R}$. This data structure must support the following operations: insertion and deletion of rectangles, orthogonal range queries for finding those rectangles R which intersect a new box B_n. We have evaluated two different structures: a kd-tree and a uniform grid for space-partitioning. In our application context, the uniform grid (with an adequate grid length) showed a better performance. Profiling of the code revealed that 70 percent of the overall runtime is spent for the range queries using the kd-tree, and 50 percent if the uniform grid is used. The data structure for the rectangles has to support a further operation: Does a given rectangle R lie completely in the interior of the union U calculated so far? This can also be solved with an orthogonal range query by shooting a ray (starting on R) in direction of an axis to infinity and analyzing the orientation of the rectangle from $\mathcal{R}$ which is first hit.

2.3 Running time

Although the theoretical running time of the incremental algorithm is quadratic in n, its experimental performance is far better at least in our application of computing the Minkowski difference between a point set and a box. The running time strongly depends on the relation between the density of the point cloud and the size of the boxes.

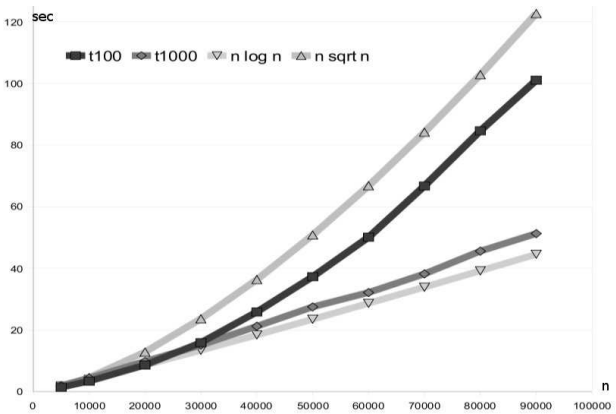


Figure 1: Runtimes of the incremental algorithm for n boxes with varying density

Figure 1 shows the runtimes on randomly generated points on a sphere with differing density. The black line indicates the runtimes for a dense point cloud in relation to the box size. The dark grey line shows a less dense point cloud. For comparison, the lines indicating $n \log n$ and $n^{3/2}$ are shown as light grey lines. The runtimes for these benchmark data sets are in general higher than for real trunk geometries. The complete preprocessing for a real trunk (≈ 20000 points) takes around 45 seconds for 7 different suitcases in 6 possible orientations. This improved runtime was possible due to the following observation: When computing the free space for stepwise increasing suitcases, we started with the smallest box S_1 and checked which container points $\mathcal{P}' \subseteq \mathcal{P}$ actually contributed to the boundary of the freespace for S_1. For

the freespace of a larger suitcase containing S_1 only the relevant points $\mathcal{P}'$ need to be considered.

3 The volumetric approach

3.1 The idea

In our second approach, the setting is again that the trunk is represented by a point cloud and the suitcases are axis-aligned 3d-boxes. Again all points lie on an integer grid and all side-lengths of a box are integers. For a new suitcase we want to check whether it still fits into the trunk and, if it does so, we want to insert it.

We now make a volumetric approach. The suitcase to be inserted next, we will call it *insert-box* in the following, is represented by its reference-point R_S. As in the previous approach, building Minkowski differences, each point of the point-cloud representing the trunk and also each box already inserted into the trunk leads to a set of boxes $B_1, \ldots, B_n$. We name $B_1, \ldots, B_n$ *offset-boxes*. The offset-boxes are stored in an octree. For testing whether the insert-box still fits into the trunk, we have to query the octree whether R_S lies outside each offset-box B_i. In other words, we have to test whether R_S lies inside the remaining freespace.

The shape of the offset-boxes B_i stored in the octree strongly depends on the shape of the insert-box. In the SAE standard we have seven different types of suitcases. Each suitcase has six different axis-aligned orientations. This leads to $7 \cdot 6 = 42$ different kinds of insert-boxes. For each kind we build an octree storing the respective offset-boxes. We test whether a given insert-box still fits into the trunk by checking its octree. For packing the box into the trunk we have to update each octree by inserting the respective Minkowski difference into each tree.

3.2 Preprocessing step

In a preprocessing step we build each of the 42 leaf-oriented octrees storing the offset-boxes of the point cloud representing the trunk. As usual, a node of an octree represents an axis-aligned part of the configuration space. We will call this part a *node-box*. In a leaf we store all offset-boxes B_i which have a non-empty intersection with its node-box.

We first determine an axis aligned bounding box (AABB) containing all $B_1, \ldots, B_n$. For robustness reasons we extend the AABB such that the length of each edge equals the same power of two. The root-node of the octree gets the AABB as its node-box. We keep on making every node an inner-node by adding its eight child-nodes until for a node one of the following conditions hold:

1) The edge length of the node-box is equal to one.
2) The node-box contains no freespace.
3) The number of offset-boxes intersecting the node-box is smaller than a constant k and the freespace in the node-box consists of only one connected component.

In these cases the node becomes a leaf-node. In the first case a further splitting is useless because we have already reached grid-size. Also in the second case a splitting is superfluous because no insert-box with a reference-point lying in the node-box can be inserted into the trunk. The first condition in the last case is for controlling the depth of the octree. The smaller the k the deeper the tree. The second condition, namely that the freespace inside a node-box has to be connected, is needed for the preprocessing step we will explain in the following. But first we describe how to check the criterion: We consider all offset-boxes intersecting the node-box. We partition the node-box in disjoint boxes such that each box is either completely covered by offset-boxes or completely belongs to the freespace. We call this step *computing freespace-boxes* and it is realized by cutting out the offset-boxes one by one. Now we start with one freespace-box and do a depth-first search on all adjacent freespace-boxes. If all freespace-boxes are visited, we only have one connected freespace component in the node-box, consider fig. 2.

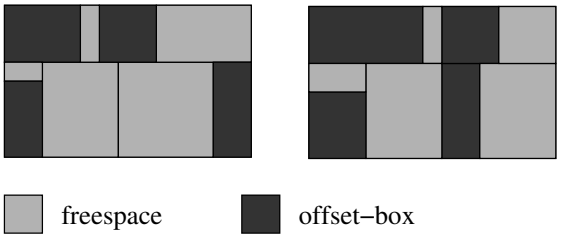


Figure 2: In the left figure the freespace is connected, in the right one it is not.

3.3 Forbidden freespace

Why do we need the last criterion for the preprocessing step? In the preprocessing step we first compute the AABB containing the trunk and insert each box from the set $\mathcal{P} \ominus \mathcal{S}$ into the octree. Now we have the following problem: There are points inside the AABB lying inside the trunk and some lying outside the trunk. More precisely, we have points inside the AABB not contained in any offset-box $B \in \mathcal{P} \ominus \mathcal{S}$ (i.e. points in the freespace of the AABB) which are inside the trunk and some which are outside. In order to be able to use the octree for our queries and insertions of suitcases later on we have to mark the freespace outside the trunk as *forbidden freespace*. Here we assume

that the freespace inside the trunk consists of one connected component and has no connection with points outside the trunk which makes sense for real trunks.

In order to find out and mark the forbidden freespace in the octree we start with a leaf of the octree which surely contains freespace inside the trunk and mark it. We omit the details on how to find this leaf. Condition 3) now ensures that if a leaf contains freespace inside the trunk, its whole freespace is inside. Using a depth-first search we consider all neighboring leafs. Only in case the new leaf has freespace connected with the freespace of the old leaf-node, the new leaf is also marked and processed further on. At the end all leafs which are not marked only contain forbidden freespace and are not considered any more.

The preprocessing step is finished and we describe how to perform the different queries using the octrees.

3.4 Computing the volume of the freespace

As mentioned before, an interesting question is to determine how much freespace we still have. In general, the freespace is different for every octree. For computing the volume of the freespace for one octree we consider all its leafs. For each leaf we compute its freespace-boxes and sum up over all volumes of the freespace-boxes:

$$\text{volume } (F) \;=\; \sum_{\substack{\text{leaves } l \\ \text{in octree}}} \;\; \sum_{\substack{\text{freespace-} \\ \text{boxes } b \text{ of } l}} \text{volume } (b).$$

3.5 Checking a reference point

To check whether an insert-box can be inserted into the trunk we consider its octree. We make a range query of its reference point in order to determine the leaf of the octree containing the reference point. If the leaf is marked as containing no freespace we are done, the insert-box is not admissible. Otherwise we test the reference point against all offset-boxes stored in the leaf which are less than k. If it is contained in none of the boxes the insert-box is admissible.

3.6 Inserting a box

For inserting a box into the data structure we have to insert its respective Minkowski difference into each octree. Therefore we first search in each octree the leafs intersecting the new offset-box. For each leaf we store the new offset-box in its list of intersecting offset-boxes. We have to check whether the leaf still contains freespace (this is done by computing freespace-boxes) and if not we have to mark it. It also can happen that now the number of offset-boxes stored exceeds the number $k-1$. If additionally the node-box is bigger than unit-size, the leaf becomes an inner node and the eight child-nodes have to be computed.

3.7 Removing a box

For removing a box we search in each octree all leaves containing the box. The box is deleted from the list of intersecting offset-boxes. If the leaf was marked containing no freespace, we have to check whether the removal causes freespace in the leaf. If the deletion has the effect that the number of different offset-boxes stored in the leafs of an inner node becomes less than k, the leafs are deleted and the inner node becomes a leaf.

4 Summary

We have presented two different algorithms to calculate the union of axis-aligned boxes and to determine available free space between these boxes. Both approaches are implemented. The first one shows a significantly better running time behaviour. This is why this approach is integrated into an actual packing algorithm. In the second algorithm the most time-consuming procedure is the computation of the freespace boxes which is called several times in every insertion and deletion step. One idea for further improvement is to relax condition 2) in order to reduce the number of times the procedure is called.

References

[1] Ernst Althaus, Tobias Baumann, Elmar Schömer, and Kai Werth. Trunk packing revisited. In Camil Demetrescu, editor, *WEA*, volume 4525 of *Lecture Notes in Computer Science*, pages 420–432. Springer, 2007.

[2] Karen Daniels and Victor Milenkovic. Multiple translational containment: Approximate and exact algorithms. In *SODA '95: Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 205–214, Philadelphia, PA, USA, 1995. Society for Industrial and Applied Mathematics.

[3] Victor Klee. Can the measure of $\cup[a_i, b_i]$ be computed in less than $o(n \log n)$ steps? *American Mathematical Monthly*, 84:284–285, 1977.

[4] Society of Automotive Engineers. SAE J1100, Motor Vehicle Dimensions, February 2001.

[5] Mark H. Overmars and Chee-Keng Yap. New upper bounds in klee's measure problem (extended abstract). In *IEEE Symposium on Foundations of Computer Science*, pages 550–556, 1988.

On Computing the Vertex Centroid of a Polytope*

Hans Raj Tiwary†

Abstract

Let $\mathcal{P}$ be an $\mathcal{H}$-polytope in $\mathbb{R}^d$ with vertex set V. The vertex centroid is defined as the average of the vertices. We prove that computing the vertex centroid of an $\mathcal{H}$-polytope is #P-hard. Moreover, we show that checking whether the vertex centroid lies in a given halfspace is also #P-hard for $\mathcal{H}$-polytopes. We also consider the problem of approximating the vertex centroid by finding a point within an ϵ distance from it and prove it to be #P-easy by showing that given an oracle for counting the number of vertices of an $\mathcal{H}$-polytope, one can approximate the vertex centroid in polynomial time. Finally, we show that any algorithm approximating the vertex centroid to *any* "sufficiently" non-trivial (for example constant) distance, can be used to construct a fully polynomial approximation scheme for approximating the centroid and also an output-sensitive polynomial algorithm for the Vertex Enumeration problem.

1 Introduction

Let $\mathcal{P}$ be an $\mathcal{H}$-polytope in $\mathbb{R}^d$ with vertex set V. Various notions try to capture the essence of a "center" of a polytope. Perhaps the most popular notion is that of the center of gravity of $\mathcal{P}$. Recently Rademacher proved that computing the center of gravity of a polytope is #P-hard [6]. The proof essentially rests on the fact that the center of gravity captures the volume of a polytope perfectly and that computing the volume of a polytope is #P-hard [3]. Also, polynomial algorithms exist that approximate the volume of a polytope within any arbitrary factor [4]. It is also easy to see that approximating the center of gravity can be done by simply sampling random points from the polytope, the number of samples depending polynomially on the desired approximation (See Algorithm 5.8 of [4]).

In this paper we study a variant of the notion of "center" defined as the centroid (average) of the vertices of P. Despite being quite a natural property of polytopes, this variant seems to have received very little attention both from theoretical and computational perspectives. Throughout this paper we will refer to the vertex centroid just as centroid. The reader should note that in popular literature the word centroid refers more commonly to the center of gravity. We nevertheless use the same terminology for simplicity of language. Our motivation for studying the centroid stems from the fact that the centroid encodes the number of vertices of a polytope. As we will see, this also makes computing the centroid hard.

The parallels between centroid and the center of gravity of a polytope mimic the parallels between the volume and the number of vertices of a polytope. Computing the volume and the number of vertices are both #P-complete ([2, 3, 5]) and so are the problems of computing the corresponding vertices ([6], Theorem 1). The volume can be approximated quite well but approximating the number of vertices of a polytope is an interesting open problem. Similarly, the center of gravity can be approximated quite well but (as we will see in this paper) obtaining a polynomial algorithm for approximating the centroid would be a very interesting achievement.

The problem of enumerating vertices of an $\mathcal{H}$-polytope has been studied for a long time. However, in spite of years of research it is neither known to be hard nor is there an output sensitive polynomial algorithm for it. A problem that is polynomially equivalent to the Vertex Enumeration problem is to decide if a given list of vertices of an $\mathcal{H}$-polytope is complete [1]. In this paper we show that any algorithm that approximates the centroid of an arbitrary polytope to any "sufficiently" non-trivial distance can be used to obtain and output sensitive polynomial algorithm for the Vertex Enumeration problem.

The main results of this paper are the following:

- Computing the centroid of an $\mathcal{H}$-polytope is #P-hard.
- Deciding whether the centroid of an $\mathcal{H}$-polytope lies in a halfspace remains #P-hard.
- Approximating the centroid of an $\mathcal{H}$-polytope is #P-easy.
- Any algorithm approximating the centroid within a distance $d^{\frac{1}{2}-\delta}$ can be used to obtain a fully polynomial approximation scheme for the centroid approximation problem and also an output sensitive polynomial algorithm for the Vertex Enumeration problem.

*This work was done when the author was supported by Graduiertenkolleg fellowship for PhD studies provided by Deutsche Forschungsgemeinschaft.

†FR Informatik, Universität des Saarlandes, D-66123, Saarbrücken, Germany, hansraj@cs.uni-sb.de

2 Results

The most natural computational question regarding the centroid of a polytope is whether we can compute the centroid efficiently. The problem is trivial if the input polytope is presented by its vertices. So we will assume that the polytope is presented by its facets. Perhaps not surprisingly, computing the centroid of an $\mathcal{H}$-polytope turns out be #P-hard. We prove this by showing that computing the centroid of an $\mathcal{H}$-polytope amounts to counting the vertices of the same polytope, a problem known to be #P-hard.

Theorem 1 *Given an $\mathcal{H}$-polytope $\mathcal{P} \subset \mathbb{R}^d$, it is #P-hard to compute its centroid $c(\mathcal{P})$.*

Proof. We embed $\mathcal{P}$ in $\mathbb{R}^{d+1}$ by putting a copy of $\mathcal{P}$ in the hyperplane $x_{d+1} = 1$ and making a pyramid with the base $\mathcal{P}$ and apex at origin. Call this new polytope $\mathcal{Q}$. Treating the direction of the positive x_{d+1}-axis as up, it is easy to see that the centroid of the new polytope lies at a height $1 - \frac{1}{n+1}$ iff the number of vertices of $\mathcal{P}$ is n. Thus any algorithm for computing the centroid can be run on $\mathcal{Q}$ and the number of vertices of $\mathcal{P}$ can be read off the $(d+1)$-st coordinate. □

Suppose, instead, that one does not want to compute the centroid exactly but is just interested in knowing whether the centroid lies to the left or to the right of a given arbitrary hyperplane. This problem turns out to be hard too, and it is not difficult to see why.

Theorem 2 *Given an $\mathcal{H}$-polytope $\mathcal{P} \subset \mathbb{R}^d$ and a hyperplane $h = \{a \cdot x = b\}$, it is #P-hard to decide whether $a \cdot c(\mathcal{P}) \leq b$.*

Proof. Consider the embedding and the direction pointing upwards as used in the proof of Theorem 1. Given an oracle answering sidedness queries for the centroid and any arbitrary hyperplane, one can perform a binary search on the height of the centroid and locate the exact height. The number of queries needed is only logarithmic in the number of vertices of $\mathcal{P}$. □

As stated before, even though computing the gravitational centroid of a polytope exactly is #P-hard, it can be approximated to any precision by random sampling. Now we consider the problem of similarly approximating the vertex centroid of an $\mathcal{H}$-polytope. Let $dist(x, y)$ denote the Euclidean distance between two points $x, y \in \mathbb{R}^d$. We are interested in the following problem:
Input: $\mathcal{H}$-polytope $P \subset \mathbb{R}^d$ and a real number $\epsilon > 0$.
Output: $p \in \mathbb{R}^d$ such that $dist(c(P), p) \leq \epsilon$.

We would like an algorithm for this problem that runs in time polynomial in the number of facets of P, the dimension d and $\frac{1}{\epsilon}$. Clearly, such an algorithm would be very useful because if such an algorithm is found then it can be used to test whether a polytope described by m facets has more than n vertices, in time polynomial in m, n and the dimension d of the polytope by setting $\epsilon < \frac{1}{2n(n+1)}$. This in turn would yield an algorithm that computes the number of vertices n of a d-dimensional polytope with m facets, in time polynomial in m, n and d. As stated before, a problem that is polynomially equivalent to the Vertex Enumeration problem is to decide if a given list of vertices of an $\mathcal{H}$-polytope is complete [1]. Clearly then, a polynomial approximation scheme for the centroid problem would yield an output-sensitive polynomial algorithm for the Vertex Enumeration problem.

Also, the problem of approximating the centroid is not so interesting if we allow polytopes that contain an arbitrarily large ball, since this would allow one to use an algorithm for approximating the centroid with *any* guarantee to obtain another algorithm with an arbitrary guarantee by simply scaling the input polytope appropriately, running the given algorithm and scale back. So we will assume that the polytope is contained in the unit hypercube in $\mathbb{R}^d$.

Now we prove that the problem of approximating the centroid is #P-easy. We do this by showing that given an algorithm that computes the number of vertices of an arbitrary polytope (a #P-complete problem), one can compute the centroid to any desired precision by making a polynomial (in $\frac{1}{\epsilon}$, the number of facets and the dimension of the polytope) number of calls to this oracle. Notice that in the approximation problem at hand, we are required to find a point within a d-ball centered at the centroid of the polytope and radius ϵ. We first modify the problem a bit by requiring to report a point that lies inside a hypercube, of side length 2ϵ, centered at the centroid of the polytope. (The hypercube has a clearly defined center of symmetry, namely its own vertex centroid.) To see why this does not essentially change the problem, note that the unit hypercube fits completely inside a d-ball with the same center and radius $\frac{\sqrt{d}}{2}$. We will call any point that is a valid output to this approximation problem, an ϵ-approximation of the centroid $c(P)$.

Given an $\mathcal{H}$-polytope P and a hyperplane $\{a \cdot x = b\}$ that intersects P in the relative interior and does not contain any vertex of P, define P_1 and P_2 as follows:

$$\begin{aligned} P_1 &= P \cap \{x | a \cdot x \leq b\} \\ P_2 &= P \cap \{x | a \cdot x \geq b\} \end{aligned}$$

Let V_1 be the common vertices of P_1 and P, and V_2 be common vertices of P_2 and P. The following lemma gives a way to obtain the ϵ-approximation of

the centroid of P from the ϵ-approximations of the centroids of V_1 and V_2.

Lemma 3 *Given P, V_1, V_2 defined as above, let n_1 and n_2 be the number of vertices in V_1 and V_2 respectively. If c_1 and c_2 are ϵ-approximations of the centroids of V_1 and V_2 respectively, then $c = \frac{n_1c_1+n_2c_2}{n_1+n_2}$ is an ϵ-approximation of the centroid c^* of P.*

Proof. Let c_{ij} be the j-th coordinate of c_i for $i \in \{1, 2\}$. Also, let c_i^* be the actual centroid of V_i with c_{ij}^* denoting the j-th coordinate of c_i^*. Since c_i approximates c_i^* within a hypercube of side-length 2ϵ, for each $j \in \{1, \cdots, d\}$ we have

$$c_{ij}^* - \epsilon \leq \quad c_{ij} \quad \leq c_{ij}^* + \epsilon$$

Also, since c^* is the centroid of P,

$$c^* = \frac{n_1c_1^* + n_2c_2^*}{n_1 + n_2}$$

Hence, for each coordinate c_j^* of c^* we have

$$\begin{array}{lcl} & \frac{n_1(c_{1j}-\epsilon)+n_2(c_{2j}-\epsilon)}{n_1+n_2} \leq c_j^* \leq \frac{n_1(c_{1j}+\epsilon)+n_2(c_{2j}+\epsilon)}{n_1+n_2} \\ \Rightarrow & \frac{n_1c_{1j}+n_2c_{2j}}{n_1+n_2} - \epsilon \leq c_j^* \leq \frac{n_1c_{1j}+n_2c_{2j}}{n_1+n_2} + \epsilon \\ \Rightarrow & c_j - \epsilon \leq c_j^* \leq c_j + \epsilon \\ \Rightarrow & c_j^* - \epsilon \leq c_j \leq c_j^* + \epsilon \end{array}$$

□

Now to obtain an approximation of the centroid, we first slice the input polytope P from left to right into $\frac{1}{\epsilon}$ slices each of thickness at most ϵ. Using standard perturbation techniques we can ensure that any vertex of the input polytope does not lie on the left or right boundary of any slice. For each slice any point in the interior gives us an ϵ-approximation of the vertices of P that are contained in that slice. We can compute the number of vertices of P lying in this slice by using the oracle for vertex computation and then using the previous Lemma we can obtain the centroid of P. Thus we have the following theorem:

Theorem 4 *Given a polytope P contained in the unit hypercube, the ϵ-approximation of the centroid of P can be computed by making a polynomial number of calls to an oracle for computing the number of vertices of a polytope.*

Now we present a bootstrapping theorem indicating that any "sufficiently" non-trivial approximation of the centroid can be used to obtain arbitrary approximations. For the notion of approximation let us revert back to the Euclidean distance function. Thus, any point x approximating the centroid c within a parameter ϵ satisfies $dist(x, c) \leq \epsilon$. As before we assume that the polytope $\mathcal{P}$ is contained in the unit hypercube. Since the polytope is thus contained in a hyperball with origin as its center and radius at most $\frac{\sqrt{d}}{2}$, any point inside $\mathcal{P}$ approximates the centroid within a factor $\sqrt{d}$. Before we make precise our notion of "sufficiently" non-trivial and present the bootstrapping theorem, some preliminaries are in order.

Lemma 5 *Suppose $(x, y), (u, u) \in \mathbb{R}^{2d}$, where $x, y, u \in \mathbb{R}^d$, then*

$$||u - \frac{x+y}{2}|| \leq \frac{||(u, u) - (x, y)||}{\sqrt{2}},$$

where $||\cdot||$ is the Euclidean norm.

The proof of the above lemma is easy and elementary, and hence we omit it here. Next, consider the product of two polytopes. Given d-dimensional polytopes $\mathcal{P}, \mathcal{Q}$ the product $\mathcal{P} \times \mathcal{Q}$ is defined as the set $\{(x, y)|x \in \mathcal{P}, y \in \mathcal{Q}\}$. It is easy to see that the number of vertices of $\mathcal{P} \times \mathcal{Q}$ is the product of the number of vertices of $\mathcal{P}$ and that of $\mathcal{Q}$, and the number of facets of $\mathcal{P} \times \mathcal{Q}$ is the sum of the number of facets of $\mathcal{P}$ and that of $\mathcal{Q}$. Moreover, the dimension of $\mathcal{P} \times \mathcal{Q}$ is the sum of the dimensions of $\mathcal{P}$ and that of $\mathcal{Q}$.

Observation 1 *If c is the centroid of a polytope P then (c, c) is the centroid of $P \times P$.*

Suppose we are given an algorithm for finding ϵ-approximation of an arbitrary polytope contained in the unit hypercube. For example, for the simple algorithm that returns any point inside the polytope, the approximation guarantee is $\frac{\sqrt{d}}{2}$. We consider similar algorithms whose approximation guarantee is a function of the ambient dimension of the polytope. Now suppose that for the given algorithm the approximation guarantee is $f(d)$. For some parameter k consider the k-fold product of P with itself $\overbrace{P \times \cdots \times P}^{k \text{ times}}$, denoted by P^k. Using the given algorithm one can find the $f(2^kd)$ approximation of P^k and using Lemma 5 one can then find the $\frac{f(2^kd)}{\sqrt{2}^k}$-approximation of P. This gives us the following bootstrapping theorem:

Theorem 6 *Suppose we are given an algorithm that computes an $\frac{\sqrt{d}}{g(d)}$-approximation for any polytope contained in the unit hypercube in polynomial time, where $g(.)$ is an unbounded monotonically increasing function. Then, one can compute an ϵ-approximation in time polynomial in the size of the polytope and $g^{-1}(\frac{1}{\epsilon})$.*

In particular, if we have an algorithm with any fixed constant approximation guarantee for finding the centroid of any polytope, then this algorithm can be used to construct a fully polynomial approximation scheme

for the general problem. In fact any algorithm with an approximation guarantee better than $d^{\frac{1}{2}-\delta}$ for any fixed $\delta > 0$ serves the purpose.

3 Concluding remarks

In this paper we studied the problem of computing the vertex centroid exactly and approximately. Although computing the centroid exactly turns out to be a hard problem, the problem of approximating the centroid remains open. We also showed via a bootstrapping theorem that any algorithm for approximating the centroid that has sufficiently non-trivial guarantee can be used to obtain a fully polynomial approximation scheme for this problem. Also, such an algorithm will give an output sensitive polynomial algorithm for the Vertex Enumeration problem for which no such algorithm is known.

References

[1] D. Avis, D. Bremner, and R. Seidel. How good are convex hull algorithms? *Comput. Geom.*, 7:265–301, 1997.

[2] M. E. Dyer. The complexity of vertex enumeration methods. *Mathematics of Operations Research*, 8(3):381–402, 1983.

[3] M. E. Dyer and A. M. Frieze. On the complexity of computing the volume of a polyhedron. *SIAM J. Comput.*, 17(5):967–974, 1988.

[4] R. Kannan, L. Lovász, and M. Simonovits. Random walks and an $O^*(n^5)$ volume algorithm for convex bodies. *Random Structures and Algorithms*, 11(1):1–50, December 1998.

[5] N. Linial. Hard enumeration problems in geometry and combinatorics. *SIAM J. Algebraic Discrete Methods*, 7(2):331–335, 1986.

[6] L. Rademacher. Approximating the centroid is hard. In *Symposium on Computational Geometry*, pages 302–305, 2007.

Space-Filling Curve Properties for Efficient Spatial Index Structures

Herman Haverkort* Freek van Walderveen*

Abstract

For the application of space-filling curves to the creation of efficient indexes on spatial objects, we develop methods for assessing their effectiveness and provide new curves that lead to better query efficiency of created indexes.

By improving and completing earlier assessments of the quality of orderings based on space-filling curves, we try to give better theoretical background for choosing the right curve for the right application.

1 Introduction

A space-filling curve is a continuous, surjective mapping from $\mathbb{R}$ to $\mathbb{R}^d$. It was not always clear that such a mapping would exist for $d > 1$, but in the late 19th century Peano showed that it is possible for $d = 2$ and $d = 3$ [16]. Since then, quite a number of space-filling curves have appeared in the literature, but during the early days they were primarily seen as a mathematical curiosity.

Today however, space-filling curves are applied in areas as diverse as load balancing for grid computing, colour space dimension reduction, small antenna design, and the creation of spatial data indexes [10]. In the remainder of this paper, we will mainly focus on the application of space-filling curves to the creation of query-efficient spatial data indexes and in particular R-trees.

R-trees An R-tree is a data structure designed for storing spatial objects, or more specifically their bounding boxes, in external memory. The primary goal of an R-tree is to quickly answer spatial queries on large sets of objects. For the purposes of this paper, the structure of the tree itself is not very important. In practice, the query time of an R-tree is dominated by the number of leaves retrieved. We will therefore only look at the leaf level of the tree. Each leaf stores the bounding boxes of a number of objects, references to those objects, and the bounding box of all objects in this leaf, used for querying. Note that an R-tree is not uniquely defined by a set of input objects. Any distribution of the objects over the leaves may be used as basis for an R-tree, as long as each object is present in exactly one leaf, and each leaf fits in one page of the external memory. We denote the number of objects that fit in a page by B. Generally, B is also referred to as the *fan-out* of the tree nodes. One way of making the distribution is by somehow ordering the input objects along a space-filling curve (more details later) and then putting each next group of B items together in a leaf (see for example Figure 1).

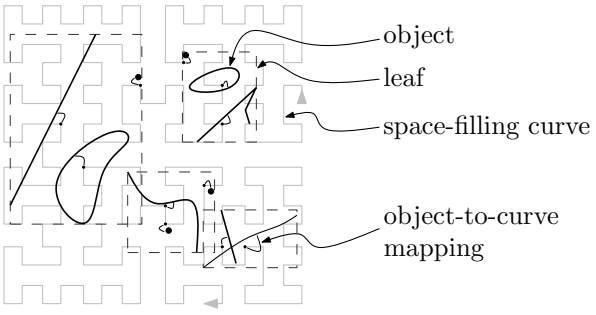


Figure 1: Leaves of an R-tree with $B = 3$

When querying these R-trees, we report all objects that intersect a given query window by checking each leaf whose bounding box intersects this query window. Thus, leaves with smaller bounding boxes have less chance of needing to be retrieved from slow external memory (for example a hard disk needing 10 ms for each seek). Using good space-filling curves that make us fill each leaf with objects that lie close to each other and thus have a small bounding box, will therefore result in better query performance. The question thus arising is: what makes a good space-filling curve?

Curves All space-filling curves considered in this paper can be constructed using a geometrical recursion scheme. See for example GP order in Figure 2, which gives a representation of the original curve by Giuseppe Peano. For convenience, we only describe the behaviour of the curves within a unit region (in this case the unit square). The first order approximation of the curve is given by the ordering of the nine subsquares, indicated by sequence numbers. Second and higher order approximations are obtained by appropriately rotating and mirroring copies of the lower order curve as indicated by the R shapes. We will call this curve *GP* instead of *Peano* to avoid confusion with other curves that have also been referred to as the Peano curve by other authors.

*Department of Mathematics and Computer Science, Eindhoven University of Technology, cs.herman@haverkort.net, freek@vanwal.nl

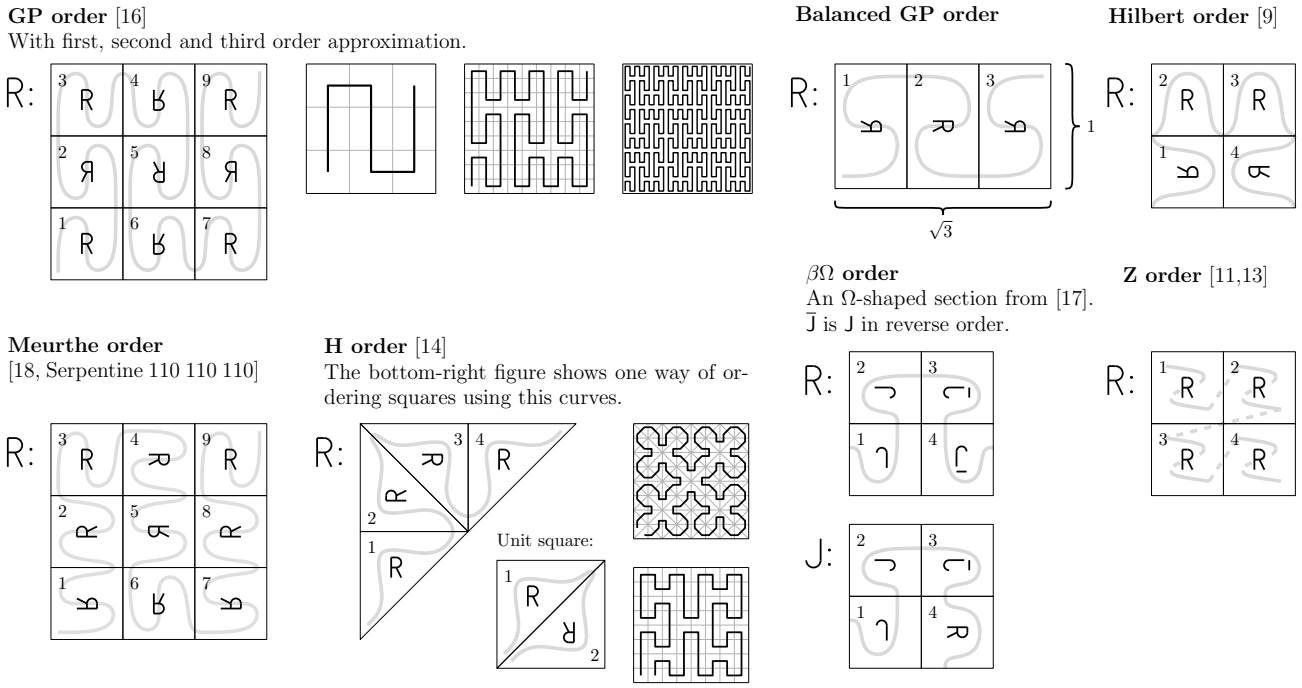


Figure 2: Space-filling curve definitions.

Although a real space-filling curve is the limit of such a recursive process, for the purposes of this paper we will only consider the process of approximating these curves by polylines and more specifically the ordering of the subsquares imposed by the curves.

A selection of the curves considered in our study is shown in Figure 2, together with their usual reference. The quality measures that we will consider for these curves will be detailed later.

Our results We propose new measures for assessing the quality of space-filling curves in their application to R-tree index creation. Bounds on these and other measures are presented for a number of well- and new or less-well-known space-filling curves. A number of these bounds improve upon results found earlier in literature. Moreover, we show that our balanced version of the GP order gives better results than the original GP order in most measures considered (and in some cases even the best).

We show that in the case of R-tree indexes on rectangles—indexed using four coordinates such as x and y coordinate of the centre, width and height—new four-dimensional Hilbert-like space-filling curves lead to better query efficiency. For these curves, we show that they exhibit a certain nice property, namely that they reduce to the well-known two-dimensional Hilbert curve if the width and height of the rectangles is zero.

2 Quality measures for curves ordering point data

We start with a simple case: ordering spatial objects that are in fact points in two dimensions. As discussed in the introduction, we are looking for space-filling curves that ensure that objects lying close to each other on the curves have small joint bounding boxes. In the literature, quite some work can be found on measures related to the proximity of points on a curve. For example, notable results were found for the following measure [7] of so called *curve-to-plane locality*

$$\lim_{n\to\infty} \max_{1\le i<j\le n} \frac{d_p(C(i),C(j))^2}{(j-i)/n}$$

where i and j are integers, $C(i)$ is the position of the ith subregion along the curve in a subdivision of the unit region into n subregions, and $d_p(P,Q)$ is the L_p distance between P and Q, thus $d_p(P,Q) = ((P_x-Q_x)^p+(P_y-Q_y)^p)^{1/p}$. We will call this measure WD_p, for Worst-case Dilation as it indicates for points that lie close on the curve how far from each other they might get in the plane. Niedermeier et al. [14] show that for any plane-filling curve, $\mathrm{WD}_1 \ge 6\frac{1}{2}$, $\mathrm{WD}_2 \ge 3\frac{1}{2}$ and $\mathrm{WD}_\infty \ge 3\frac{1}{2}$. Furthermore, several results are known for these measures regarding Hilbert order [1, 3, 7, 15, 5], GP order [12] and H order [14] (see Table 1).

Curve-to-plane locality may however not give the best prediction for bounding box areas. Therefore we propose the Worst-case Bounding-box Area measure.

$$\mathrm{WBA} = \lim_{n\to\infty} \max_{1\le i<j\le n} \frac{\mathrm{area}(\mathrm{bbox}(C(i,j)))}{\mathrm{area}(C(i,j))}$$

Order	WD_∞	WD_2	WD_1	WBA	RBA
GP	8	8	$10^2/_3$	**2.00**	1.42
Bal. GP	**4.62**	**4.62**	**8.62**	**2.00**	1.42
Meurthe	**5.33**	**5.67**	**10.67**	**2.50**	1.39
Hilbert	6	6	9	**2.40**	1.41
$\beta\Omega$	**5.00**	**5.00**	**9.00**	**2.22**	1.40
H	4	4	8	**3.00**	1.69
Z	∞	∞	∞	∞	2.86

Table 1: Bounds for different measures and curves. New bounds in bold computed to the indicated precision.

where $C(i,j)$ is the union of all subregions between i and j (inclusive), bbox(r) is the bounding box of region r and area(r) is the area of region r. Thus, for an approximation of curve C filling a grid of n squares, area$(C(i,j)) = (j-i+1)/n$.

We have the following

Lemma 1 *Any recursively defined curve filling triangles or a regular grid of axis-parallel rectangles satisfies WBA* ≥ 2.

(Proof omitted from this extended abstract.)

To assess the predictive value of the WD and WBA measures for the size of bounding boxes in practice, we generated 50 sets of points uniformly distributed in the unit region. The size of each set was chosen randomly between 150,000 and 5,400,000. The points were packed into groups of 1000 along the curves, and we measured the total area of the bounding boxes of the groups. Table 1 lists the results of this Random Bounding-box Area (RBA) experiment, as well as results for the other measures. The latter were obtained using an algorithm that approximates the values of the measures for the curves by iteratively finding better upper and lower bounds (details omitted from this abstract).

We can see that although H order performs best on the classical measures, it performs worst (when ignoring the obviously incompetitive Z order) on RBA, well predicted by WBA. Meurthe order on the other hand, having the best value for the RBA measure is not among the best regarding WBA. Thus, it would be interesting to see whether we can either find a better averaging metric instead of the current random one, or a theoretical (worst-case or other) measure that has better predictive value. Furthermore, it might be interesting to investigate if other ways of taking bounding boxes, such as using rotated axes, lead to better performance of particular curves.

Finally, we see that the Balanced GP order, which is really a horizontal stretching of GP order by $\sqrt{3}$, achieves much better results for the WD measures (note that such a scaling is not interesting for the other curves as they have more symmetric behaviour).

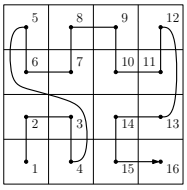


Figure 3: Second order approximation of the four-dimensional Hilbert-like curve by Alber and Niedermeier [1] when restricted to points with $x_{\mathrm{mn}} = x_{\mathrm{mx}}$ and $y_{\mathrm{mn}} = y_{\mathrm{mx}}$.

3 Four-dimensional curves ordering rectangle data

Because R-trees are built using rectangular bounding boxes (or even n-dimensional ones), it is also—and perhaps even more—interesting to investigate orderings on four (or more) dimensions. Kamel and Faloutsos [10] consider the use of a four-dimensional version of the Hilbert space filling curve for ordering rectangles. They consider two variants of this approach, which map an object's bounding box $[x_{\mathrm{mn}}, x_{\mathrm{mx}}] \times [y_{\mathrm{mn}}, y_{\mathrm{mx}}]$ to either a four-dimensional point $(x_{\mathrm{mn}}, y_{\mathrm{mn}}, x_{\mathrm{mx}}, y_{\mathrm{mx}})$, dubbed the *4D-xy* mapping, or a four-dimensional point (c_x, c_y, d_x, d_y), where $c_x = \frac{1}{2}(x_{\mathrm{mn}} + x_{\mathrm{mx}})$, $c_y = \frac{1}{2}(y_{\mathrm{mn}} + y_{\mathrm{mx}})$, $d_x = x_{\mathrm{mx}} - x_{\mathrm{mn}}$, and $d_y = y_{\mathrm{mx}} - y_{\mathrm{mn}}$, dubbed the *4D-cd* mapping.

Kamel and Faloutsos compare the performance of R-trees based on these approaches to the 2D Hilbert order based only on the centre point of each rectangle. Surprisingly, this last method performs best in their experiments. Later experiments [2] agree on this when the data is relatively close to point data. However on more extreme, artificial data sets it was shown that the 4D-xy approach easily outperforms 2D Hilbert order. Apparently the 4D orders do not closely resemble the 2D order when near-point data is supplied.

As shown by Alber and Niedermeier [1], there is no one true four-dimensional version of the Hilbert order. In fact there are very many 4D orders that one could classify as having the "Hilbert property". Alber and Niedermeier also propose a formalism for writing down higher dimensional Hilbert-like curves, based on permutations of hyperquadrants and, as an example of this generalization, give the constructing elements for a four-dimensional curve. Another four-dimensional curve that appears in the literature is based on work by Butz [4] and implemented by Doug Moore.

Both of these curves do not follow the original two-dimensional Hilbert curve when we use them to order objects for which $x_{\mathrm{mn}} = x_{\mathrm{mx}}$ and $y_{\mathrm{mn}} = y_{\mathrm{mx}}$, see Figure 3 for an example. Moreover, the resulting orders make jumps in the grid, eventually resulting in worse query performance of R-trees based on them.

A natural question is thus, are there any four-

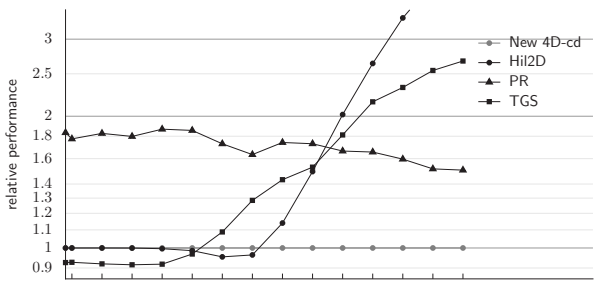


Figure 4: Number of leaves accessed by different R-trees on data with only small (left) to larger (right) rectangles, normalized by dividing by the performance of the newly proposed curve.

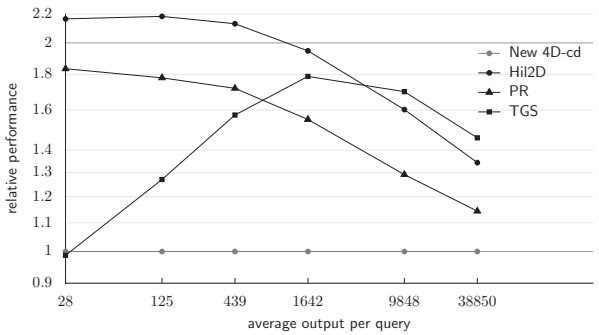


Figure 5: Relative number of leaves accessed by different R-trees on a data set taken from VLSI design for different query sizes.

dimensional Hilbert-like space-filling curves that do follow the original Hilbert curve when "projected" down to two dimensions? Using an automated search procedure, we have found 218 such curves for the 4D-cd case in a representative part of the search space (which is much too large to be scanned completely). From these curves, we picked the one that performed best on some small experiments. Then we ran experiments on large artificial and real-life data sets and compared the query performance of R-trees based on this curve to the some of the best performing R-trees currently known [2, 6, 10]. Some results can be found in Figures 4 and 5. For a more detailed discussion of these and other experiments, we refer to our manuscript [8]. There, we conclude that our new 4D-cd based ordering matches the performance of the 2D Hilbert curve for data sets with only small rectangles. For data sets with larger rectangles however, the new ordering matches or outperforms any previously known R-tree.

4 Conclusion

Space-filling curves are (still) both theoretically interesting and useful in practice. We have shown that the efficiency of R-tree indexes can be improved by carefully picking the space-filling curve to be used. A number of theoretical results regarding quality measures were given, but there is still a lot to explore.

References

[1] J. Alber and R. Niedermeier. On multidimensional curves with Hilbert property. *Theory of Computing Systems*, 33(4):295–312, 2000.

[2] L. Arge, M. de Berg, H. J. Haverkort, and K. Yi. The Priority R-tree: a practically efficient and worst-case optimal R-tree. In *ACM SIGMOD conference on Management of Data*, pp 347–358, New York, 2004.

[3] K. E. Bauman. The dilation factor of the Peano-Hilbert curve. *Math. Notes*, 80(5):609–620, 2006.

[4] A. R. Butz. Alternative algorithm for Hilbert's space-filling curve. *IEEE Transactions on Computers*, pp 424–426, 1971.

[5] G. Chochia, M. Cole, and T. Heywood. Implementing the hierarchical PRAM on the 2D mesh: Analyses and experiments. In *Symp. on Parallel and Distributed Processing*, pp 587–595, 1995.

[6] Y. J. García, M. A. López, and S. T. Leutenegger. A greedy algorithm for bulk loading R-trees. In *ACM Symp. on Advances in GIS*, pp 163–164, 1998.

[7] C. Gotsman and M. Lindenbaum. On the metric properties of discrete space-filling curves. *IEEE Trans. Image Processing*, 5(5):794–797, 1996.

[8] H. Haverkort and F. van Walderveen. Bulk loading R-trees with four-dimensional space-filling curves. Manuscript, 2008.

[9] D. Hilbert. Über die stetige Abbildung einer Linie auf ein Flächenstück. *Math. Ann.*, 38(3):459–460, 1891.

[10] I. Kamel and C. Faloutsos. On packing R-trees. In *Conf. on Inf. and Knowl. Man.*, pp 490–499, 1993.

[11] H. L. Lebesgue. *Leçons sur l'intégration et la recherche des fonctions primitives*, pp 44–45. Gauthier-Villars, 1904.

[12] U. von Luxburg. Lokalitätsmaße von Peanokurven. Stud. project report, Universität Tübingen, 1998.

[13] G. M. Morton. A computer oriented geodetic data base and a new technique in file sequencing. Technical report, IBM, Ottawa, 1966.

[14] R. Niedermeier, K. Reinhardt, and P. Sanders. Towards optimal locality in mesh-indexings. *Discrete Applied Mathematics*, 117:211–237, 2002.

[15] R. Niedermeier and P. Sanders. On the Manhattan-distance between points on space-filling mesh-indexings. Technical Report IB 18/96, Karlsruhe University, Dept. of Computer Science, 1996.

[16] G. Peano. Sur une courbe, qui remplit toute une aire plane. *Math. Ann.*, 36(1):157–160, 1890.

[17] J.-M. Wierum. Definition of a new circular space-filling curve: $\beta\Omega$-indexing. TR-001-02, Paderborn Center for Parallel Computing (PC^2), 2002.

[18] W. Wunderlich. Über Peano-Kurven. *Elemente der Mathematik*, 28(1):1–10, 1973.

Optimizing Active Ranges for Consistent Dynamic Map Labeling

Ken Been* Martin Nöllenburg† Sheung-Hung Poon‡ Alexander Wolff§

Abstract

Map labeling encounters unique issues in the context of dynamic maps with continuous zooming and panning—an application with increasing practical importance. In *consistent* dynamic map labeling, distracting behavior such as popping and jumping is avoided. In our model a *dynamic label placement* is a continuous function that assigns a 2d-label to each scale. This defines a 3d-solid, with scale as the third dimension. To avoid popping, we truncate each solid to a *single* scale range, called its *active range*. This range corresponds to the interval of scales at which the label is visible. The *active range optimization (ARO)* problem is to select active ranges so that no two truncated solids overlap and the sum of the active ranges is maximized. We show that the ARO problem is NP-complete, even for quite simple solid shapes, and we present constant-factor approximations for different variants of the problem.

1 Introduction

Recent years have seen tremendous improvements in Internet-based, geographic visualization systems that provide continuous zooming and panning (e.g., Google Earth), but relatively little attention has been paid to special issues faced by map labeling in such contexts. In addition to the need for interactive speed, several desiderata for a *consistent dynamic labeling* were identified in [1]: labels do not pop in and out or jump (suddenly change position or size) during panning and zooming, and the labeling is a function of scale and view area—it does not depend on the user's navigation history.

Model. We adapt the following labeling model from [1], with slight changes. In *static labeling*, the key operations are selection and placement—select a subset of the labels that can be placed without overlap. A static placement of a label L is a transformation π^L, composed of translation, rotation, and dilation, that takes L's canonical shape into world coordinates. Once all labels are placed, a viewing transformation takes world coordinates to map coordinates.

In *dynamic labeling* we take scale as an additional dimension. As with [1, 4], we define scale as the inverse of cartographic scale, so that it increases when zooming out. A *dynamic placement* of L is a function that assigns a static placement π^L_s to each scale $s \geq 0$. The translation, rotation and dilation components of the dynamic placement must each be continuous functions of scale. This eliminates jumping and popping during panning, and dependence on navigation history. *Dynamic selection* is similarly a Boolean function of scale. To eliminate popping during zooming we require that each label L_i, $1 \leq i \leq n$, is selected precisely on a single interval of scales, $[a_i, A_i]$, which is called the *active range* of L_i. Thus all consistency desiderata can be satisfied by adhering to this model.

Let $S_{\max}$ be a universal maximum scale for all labels. We define the *available range* of L_i to be an interval of scales, $[s_i, S_i] \subseteq [0, S_{\max}]$, in which label L_i "wants" to be selected. We require $[a_i, A_i] \subseteq [s_i, S_i]$. Since the dynamic placement is continuous with scale, $E_i = \bigcup_{s\in[s_i,S_i]} \pi^{L_i}_s(L_i)$ is a solid defined by sweeping the label shape along a continuous curve that is monotonic in scale, see Fig. 1. We call E_i the *extrusion* of L_i and $T_i = \bigcup_{s\in[a_i,A_i]} \pi^{L_i}_s(L_i)$ its *truncated extrusion*.

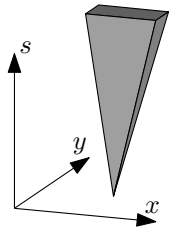


Figure 1: A dynamic label placement is a solid in world coordinates.

The extrusion shapes are determined by the label shape and the translation, rotation and dilation functions that compose the dynamic placement. We restrict our attention to certain classes of extrusions. Our labels are rectangular. For translation, we consider only *invariant point placements*, in which a particular point on the label always maps to the same location in world coordinates, so the label never "slides". Our rotation functions are constant, and yield axis-aligned labels. We consider two classes of dilation functions D^L. If $D^L(s) = bs$ for a constant $b > 0$, then label size is fixed on screen and proportional to scale in world coordinates. The solid is then a label-shaped cone with apex at $s = 0$ as in Fig. 1. With invariant point placements, the cone contains

*Computer Science Department, Yeshiva University, New York, NY, U.S.A. kbeen@yu.edu

†Fakultät für Informatik, Universität Karlsruhe, Germany. noellenburg@iti.uka.de

‡Department of Computer Science, National Tsing Hua University, Hsin-Chu, Taiwan. spoon@cs.nthu.edu.tw

§Faculteit Wiskunde en Informatica, Technische Universiteit Eindhoven, The Netherlands. http://www.win.tue.nl/~awolff

extrusion shape	ARO	dilation	approx.	running time	reference
congruent square cones	simple	bs	1/4	$O((k+n)\log^2 n)$	Theorem 4
congruent square cones		bs	1/8	$O(n\log^3 n)$	Corollary 8
arbitrary square cones		bs	1/24	$O(n\log^3 n)$	Theorem 7
segments of congruent square cones	general	bs	1/4	$O((k+n)\log^2 n)$	Theorem 4
congruent frusta		$bs+c$	$1/(4W)$	$O(n^4)$	Theorem 3

Table 1: Results attained in this paper, where k is the number of pairwise intersections between extrusions and W is the width ratio of top over bottom side.

the vertical line through its apex. The cone might be symmetric to that line (e.g., for labeling a region) or might have a vertical side incident to it (e.g., for labeling a point). Secondly, we consider, in a more general setting, functions of the form $D^L(s) = bs + c$ for constants $b > 0$ and $c \neq 0$. The solid in this case is a portion of a cone with apex at $-c/b$.

Objective. Let $\mathcal{E}$ denote the set of all extrusions, and assume we are given an available range for each. For a set $\mathcal{T}$ of truncated extrusions, define $H(\mathcal{T}) = \sum_{i=1}^{n}(A_i - a_i)$ to be the *total active range height*. This is the same as integrating over all scales the function $f(s)$ that counts the number of labels selected at scale s. The *(general) active range optimization (ARO)* problem is to choose the active ranges so as to maximize H, subject to the constraint that no two truncated extrusions overlap. This is the dynamic analogue of placing the maximum number of labels without overlap in the static case. We call any set of active ranges that correspond to non-overlapping truncated extrusions a *solution*. It is of both theoretical and practical interest to also consider a version of the problem in which $[s_i, S_i] = [0, S_{\max}]$ and $a_i = 0$ for all i. We call this variant of ARO *simple*. In this version all labels want to be selected at all scales, and a label is never deselected when zooming in.

Already the simple ARO problem is NP-complete. Table 1 summarizes the approximation results obtained in this paper. In the full version we also consider 1d-labels, which are segments on the x-axis. The 1d-problem can be seen as a scheduling problem with geometric constraints and is closely related to geometric maximum independent set problems.

Previous work. Map labeling has been the focus of extensive algorithmic investigation, see the map-labeling bibliography [5]. However, the majority of the research efforts cover static labeling. For dynamic labeling, Petzold et al. [2, 3] use a preprocessing phase to generate a data structure that is searched during interaction to produce a labeling for the current scale and view area. Poon and Shin [4] build a hierarchy of precomputed solutions, and interpolation between these produces a solution for any scale. Neither of these approaches satisfies the consistency desiderata. In addition to introducing consistency for dynamic map labeling, Been et al. [1] show that simple ARO is NP-complete for star-shaped labels, and implement a simple heuristic solution in a working system.

2 Complexity

Already the simple ARO problem for congruent square cones as extrusion shape is NP-complete. The proof is by reduction from Planar3SAT using 3d-gadgets. We omit it here due to space constraints.

Theorem 1 *Simple ARO with proportional dilation is NP-complete, i.e., given a real $K > 0$ and a set $\{E_1, \ldots, E_n\}$ of congruent square cones, it is NP-complete to decide whether there is a set of truncated extrusions $\mathcal{T} = \{T_1, \ldots, T_n\}$ with $T_1 \subseteq E_1, \ldots, T_n \subseteq E_n$ and $H(\mathcal{T}) \geq K$.*

3 Approximation algorithms

In this section we give two algorithms that yield constant-factor approximations for a number of different variants of the ARO problem. The first algorithm in Sect. 3.1 is based on sweeping the extrusions from top to bottom and the second one in Sect. 3.2 is a level-based greedy algorithm. Due to space constraints we omit the proofs of the running times.

3.1 Top-to-bottom fill-down sweep

Algorithm 1 below is based on the idea to sweep down over the extrusions in $\mathcal{E}$, and if $E_i \in \mathcal{E}$ is selected at some height s, we "fill" E_i from s down to its bottom—i.e., we set $[a_i, A_i] = [s_i, s]$. Thus we have $a_i = s_i$ for every E_i that contributes to the objective function H at all.

Say that E_i is *available* if its available range includes the current sweep scale s, and *active* if its active range has already been set and covers s. We are interested in event points at which the conflict graph over the available extrusions changes. This happens at each S_i and s_i, and with some extrusion shapes it also happens at additional heights. If E_i and E_j are both available at s and at $s' > s$, and they intersect

at s' but not at s, then let s_{ij} refer to the lowest scale at which they intersect. Let k be the number of s_{ij} events over $\mathcal{E}$. We make use of a subroutine, "try to pick" E_i, which means, "if E_i does not intersect the interior of any extrusion already chosen to be active at the current sweep height s, then make E_i active and set $[a_i, A_i] = [s_i, s]$".

Algorithm 1 *Top-to-bottom sweep algorithm.*
Sweep a plane from top to bottom. At each event point of type S_i, s_i, or s_{ij}, try to pick each available but inactive extrusion E_j, in non-increasing order of S_j.

The following lemma will help proving approximation factors. Let $\mathcal{A} = \{(a_i, A_i)\}$ be the solution computed by Algorithm 1. Say that E_j *blocks* E_i at scale s under a given solution if E_i and E_j overlap (i.e., their interiors intersect) at s and $s \in [a_j, A_j]$. Note that this implies that $s \notin [a_i, A_i]$. Say that two extrusions are *independent at* s if their restrictions to the horizontal plane at height s are non-overlapping.

Lemma 2 *If, for any $E \in \mathcal{E}$ and $s \geq 0$, E can block no more than c pairwise independent extrusions at s, then $\mathcal{A}$ is a $(1/c)$-approximation for the maximum total active range height of $\mathcal{E}$.*

Proof. Suppose that $E \in \mathcal{E}$ is inactive at scale s under $\mathcal{A}$. Then E must be blocked at the nearest event point above (or at) s, since otherwise it would be picked by Algorithm 1. Since the extrusion conflict graph only changes at event points, E is blocked at s. Thus, in $\mathcal{A}$, if E is inactive at any scale s then E is blocked at s.

If at any scale no extrusion can block more than c pairwise independent extrusions, and in $\mathcal{A}$ every inactive extrusion is blocked, then at any scale the number of active extrusions in an optimal solution can be no more than c times the number in $\mathcal{A}$. Integrating over all scales proves the lemma. □

Congruent frusta. The top-to-bottom nature of Algorithm 1 ensures that if a frustum E_j blocks another frustum E_i at scale s then E_i intersects a side face of E_j. The number of independent frusta that can intersect a single face depends on W, the ratio of the side length of the top face of each frustum to that of the bottom face.

Theorem 3 *Algorithm 1 computes a $1/(4W)$-approximation for the maximum total active range height of a set of n congruent frusta in $O(n^4)$ time.*

Frustal segments of congruent square cones. For congruent underlying square cones the size of all squares is the same at each scale. Thus any extrusion blocked by an extrusion E at scale s must intersect one of the four corner edges of E at s, so at most four such extrusions can be independent. The approximation factor in Theorem 4 follows from Lemma 2.

Theorem 4 *Given a set of n frustal segments of axis-aligned unit square cones, Algorithm 1 computes a $(1/4)$-approximation for the maximum total active range height in $O((n+k)\log^2 n)$ time.*

Note that simple ARO with congruent square cones is a special case of the above where each $[s_i, S_i] = [0, S_{\max}]$, so that Theorem 4 still holds in this case.

3.2 Level-based small-to-large greedy algorithm

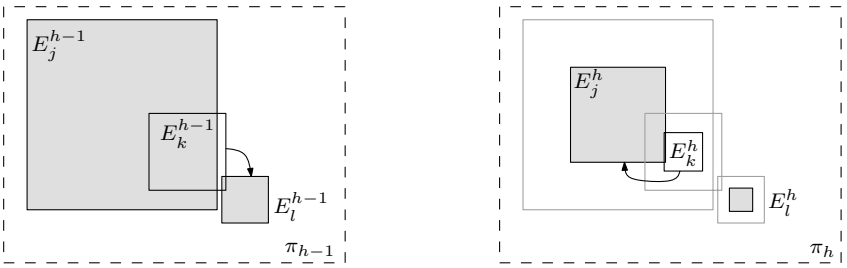
In this section we give an algorithm for simple ARO with square cones. It computes a 1/8-approximation when the cones are congruent, and a (1/24)-approximation otherwise. The algorithm intersects the given cones with $O(\log n)$ horizontal planes, starting at $S_{\max}$ and proceeding downward.

Algorithm 2 *Level-based algorithm for 3d-cones*
Initially no extrusion is active. In phase i, $i = 0, \ldots, \lceil \log n \rceil$, let π_i be the horizontal plane at scale $s = S_{\max}/2^i$. Let E^i_j be the intersection of extrusion E_j with π_i and call E^i_j *active* if E_j is already active. As long as there is an inactive object E^i_j that does not intersect any active object, choose the smallest such object $E^i_{j^\star}$ and make $E_{j^\star}$ (and $E^i_{j^\star}$) active by setting $A_{j^\star} = s$.

We first consider arbitrary square cones that are symmetric to the vertical axes passing through their apexes. When the algorithm terminates, all squares at level i that are not active must intersect an active square—they are *blocked*. We associate each blocked square E^i_j to one of the active squares in the following way: (i) If E^i_j was not blocked at the beginning of phase i but became blocked by a newly activated square E^i_k, then associate E^i_j to E^i_k. (ii) If E^i_j was blocked in the beginning of phase i then associate E^i_j to any of its blocking squares that were active at the beginning of phase i. Next, we show that the squares associated to an active square cannot be arbitrarily small.

Lemma 5 *Let E^i_j be an active square at level i with side length ℓ^i_j. Then any square associated to E^i_j has side length at least $\ell^i_j/3$ and intersects the boundary of E^i_j.*

Proof. Let E^i_k be associated to E^i_j with $\ell^i_k < \ell^i_j$. By the greedy choice of the algorithm, all squares associated to a newly active square are larger than it. This implies that E_j must have been activated at a higher level, and that E_k must have been reassigned to E_j at some level $h \leq i$. Thus, at level $h-1$ square E^{h-1}_k



(a) Squares at level $h-1$. (b) Squares at level h.

Figure 2: Intersection behavior of E_j, E_k, E_l at two consecutive levels.

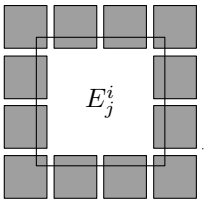


Figure 3: At most 12 independent squares intersect E_j^i.

was associated to another square E_l^{h-1}. Note that for this reassignment to take place at level h, E_j^{h-1} must have been active. Thus we know that E_j^{h-1} and E_l^{h-1} do not intersect, but they both intersect E_k^{h-1}; see Fig. 2a. At level h the reassignment takes place because E_k^h no longer intersects E_l^h but still intersects E_j^h; see Fig. 2b. Now suppose $\ell_k^h < \ell_j^h/3$. Then by going from level h to $h-1$ the side lengths of the squares are doubled and it is easy to verify that E_k^{h-1} would be contained in E_j^{h-1}, a contradiction to the fact that $E_k^{h-1} \cap E_l^{h-1} \neq \emptyset$. As $\ell_k^h \geq \ell_j^h/3$ this also holds for level i, and since E_k^{h-1} intersects the boundary of E_j^{h-1} this is also still true for level i. □

Let $\pi_{\lceil \log n \rceil + 1}$ be the plane $s = 0$, and denote the active segments of the extrusions in the optimal solution $\mathcal{S}$ and our algorithm's solution $\mathcal{A}$ between planes π_{i-1} and π_i by $\mathcal{S}_i$ and $\mathcal{A}_i$, respectively. We charge the active range height $H(\mathcal{S}_i)$ to that of $H(\mathcal{A}_{i+1})$.

Lemma 6 *For $i \in 1, \ldots, \lceil \log n \rceil - 1$ it holds that $H(\mathcal{A}_{i+1}) \geq 1/24\ H(\mathcal{S}_i)$.*

Proof. Let square E_j^i be active in $\mathcal{A}$ and consider the set $D(E_j^i)$ of squares in π_i associated to it. The squares in $D(E_j^i)$ that correspond to active extrusions in $\mathcal{S}_i$ cannot intersect each other.

By Lemma 5, all squares in $D(E_j^i)$ have side length at least $\ell_j^i/3$ and intersect the boundary of E_j^i. Thus, at most 12 of those squares can be independent in π_i and hence active in $\mathcal{S}_i$ like in Fig. 3. Now the height between levels i and $i-1$ is twice the height between levels $i+1$ and i. Hence the active height of E_j in $\mathcal{A}_{i+1}$ is at least $1/24$ times the sum of heights of active extrusions in $\mathcal{S}_i$ whose squares at level i are associated to E_j^i. It follows that $H(\mathcal{A}_{i+1}) \geq 1/24\ H(\mathcal{S}_i)$. □

Theorem 7 *Algorithm 2 computes a $(1/24)$-approximation to the maximum total active range height of a set of arbitrary square cones in $O(n \log^3 n)$ time.*

Proof. From Lemma 6, it remains to compare $H(\mathcal{S}_{\lceil \log n \rceil}) + H(\mathcal{S}_{\lceil \log n \rceil + 1})$ to $H(\mathcal{A}_{\lceil \log n \rceil + 1}) + H(\mathcal{A}_1)$. The height of $\pi_{\lceil \log n \rceil - 1}$ is at most $2S_{\max}/n$ and obviously there are at most n active cone segments in $\mathcal{S}$ below $\pi_{\lceil \log n \rceil - 1}$, so their total active range height is at most $2S_{\max}$. On the other hand, there is at least one active cone segment in $\mathcal{A}_1$ of height $S_{\max}/2$. Thus the approximation factor is indeed $1/24$. □

With congruent square cones, all squares at each level are the same size, so at most four rather than 12 independent squares can intersect a given square. A similar argument gives the following corollary.

Corollary 8 *Algorithm 2 computes a $(1/8)$-approximation to the maximum total active range height of a set of congruent square cones in $O(n \log^3 n)$ time.*

4 Conclusions

ARO is an exciting new problem inspired by interactive web-based mapping applications and we have given approximation algorithms for some variants. It remains open whether any of the problems admits a PTAS. Also, mapping applications in practice often require more complex extrusion shapes.

References

[1] K. Been, E. Daiches, and C. Yap. Dynamic map labeling. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):773–780, 2006.

[2] I. Petzold, G. Gröger, and L. Plümer. Fast screen map labeling—data-structures and algorithms. In *Proc. 23rd Internat. Cartographic Conf. (ICC'03)*, pages 288–298, Durban, South Africa, 2003.

[3] I. Petzold, L. Plümer, and M. Heber. Label placement for dynamically generated screen maps. In *Proc. 19th Internat. Cartographic Conf. (ICC'99)*, pages 893–903, 1999.

[4] S.-H. Poon and C.-S. Shin. Adaptive zooming in point set labeling. In M. Liśkiewicz and R. Reischuk, editors, *Proc. 15th Internat. Sympos. Fundam. Comput. Theory (FCT'05)*, volume 3623 of *Lecture Notes Comput. Sci.*, pages 233–244. Springer-Verlag, 2005.

[5] A. Wolff and T. Strijk. The Map-Labeling Bibliography. `http://i11www.ira.uka.de/map-labeling/bibliography`, 1996.

Order-k Triangulations of Convex Inclusion Chains in the Plane

Wael El Oraiby* Dominique Schmitt*

Abstract

Given a set V of n points in the plane, we show that there is a strong connexion between the k-sets of a convex inclusion chain of V introduced in [5] and the centroid triangulations of V defined in [8]. We also show that one of these triangulations can be constructed in $O(n \log n + k(n-k)\log^2 k)$ time.

1 Introduction

Given a finite set V of n points in the Euclidean plane (no three of them being collinear) and an integer k ($0 < k \leq n$), the k-sets of V are the subsets of k points of V that can be strictly separated from the rest by a straight line. The numbers of k-sets have been studied in various ways in computational and combinatorial geometry (see [4], [12], and [10] for some best bounds known in the plane). In [5], we have given a new invariant of the number of k-sets, in connexion with convex inclusion chains of V. Such a chain is an ordering $\mathcal{V} = (v_1, v_2, ..., v_n)$ of the points of V such that, for every $i \in \{2, ..., n\}$, v_i does not belong to the convex hull $conv(S_{i-1})$ (with $S_i = \{v_1, ..., v_i\}$, for all $i \in \{1, ..., n\}$). The set of k-sets of the convex inclusion chain $\mathcal{V}$ is then the set of distinct k-sets of $S_k, S_{k+1}, ..., S_n$. We have shown that the number of these k-sets does not depend on the chosen chain and, surprisingly, it is equal to the number of regions of the order-k Voronoi diagram of V.

Independently, while studying multivariate splines, Lyu and Snoeyink have introduced the notion of centroid triangulation [8]. It is a generalization of the order-k Delaunay diagram, which is dual to the order-k Voronoi diagram [3, 11] (note that this order-k Delaunay diagram has nothing to do with the order-k Delaunay triangulation of [6]). For $k \leq 3$, Lyu and Snoeyink have proven the correctness of there constructive definition of centroid triangulations and they have conjectured that it also holds for $k > 3$.

In this paper we establish the relation between the k-sets of the convex inclusion chains of a point set V and the centroid triangulations of V. More precisely, we show that, for all k, the centroids of the k-sets of a convex inclusion chain of V are the vertices of a centroid triangulation of V. We call this triangulation the order-k triangulation of the convex inclusion chain.

On the one hand, this result allows us to find, for all k, a family of centroid triangulations that verify the definition of Lyu and Snoeyink. On the other hand, it is a first step toward the understanding why the number of k-sets of a convex inclusion chain is equal to the number of regions of the order-k Voronoi diagram.

Finally, we show that a particular centroid triangulation can be constructed in $O(n \log n + k(n-k)\log^2 k)$ time. This improves the algorithm that follows from the constructive definition of Lyu and Snoeyink whose time complexity is at least $O(n \log n + k^2(n-k))$.

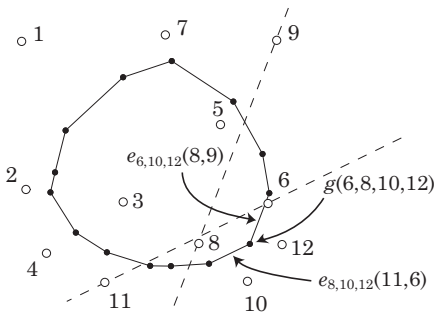


Figure 1: Edges and vertices of a 4-set polygon of 12 points.

2 k-set polygons

Given two points s and t of V, we denote by st the closed line segment with endpoints s and t oriented from s to t, by (st) the oriented straight line generated by st, and by $(st)^-$ the open half plane on the right of (st). For any subset E of the plane, we denote by $\overline{E}$ the closure of E.

Let $g^k(V)$ be the k-set polygon of V, i.e., the convex hull of the centroids of all the k-element subsets of V. Notice that, $g^1(V)$ is the convex hull $conv(V)$ of V and $g^n(V)$ is a unique point, the centroid $g(V)$ of V.

We first recall two important properties of the vertices and edges of $g^k(V)$ given by Andrzejak and Fukuda [1], and by Andrzejak and Welzl [2] (see Figure 1 for an illustration).

Proposition 1 *(i) $g(T)$ is a vertex of $g^k(V)$ if and only if T is a k-set of V.*

(ii) $g(T)g(T')$ is a counterclockwise oriented edge of $g^k(V)$ if and only if there exist two points s and t

*Laboratoire MIA, Université de Haute-Alsace, Mulhouse, France {Wael.El-Oraiby, Dominique.Schmitt}@uha.fr

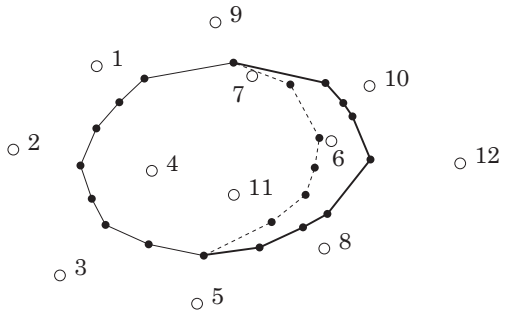


Figure 2: Construction of the 4-set polygon of $S \cup \{12\}$ from the 4-set polygon of $S = \{1, ..., 11\}$. The edges to remove are in dashed lines and the edges to create in bold lines.

of V and a subset P of $k-1$ points of V such that $T = P \cup \{s\}$, $T' = P \cup \{t\}$, and $V \cap (st)^- = P$.

From now on, any such oriented edge will be denoted by $e_P(s,t)$. Obviously, $e_P(s,t)$ is parallel to (st) and it is not difficult to see that the any line that separates the vertex $g(T)$ from the vertices of $g^k(V)$ is parallel to a line that separates T from V (and conversely).

Let S be a non empty subset of V and v be a point of $V \setminus conv(S)$, and consider the edges to remove and the ones to create when constructing $g^k(S \cup \{v\})$ from $g^k(S)$ (see Figure 2). From [5], we know that:

Proposition 2 *(i) The edges to remove are the edges $e_P(s,t)$ of $g^k(S)$ with $v \in (st)^-$. Together with their endpoints, they form a connected polygonal line $\mathcal{D}^k_{S,v}$.*

(ii) The edges to create form a connected polygonal line of at least two edges. The first (resp. last) of them (in counterclockwise direction) is of the form $e_P(s,t)$ with $t = v$ (resp. $s = v$). The other edges to create form a polygonal line $\mathcal{C}^k_{S,v}$ of edges of the form $e_P(s,t)$ with $v \in P$.

For every vertex $g(T_i)$ of $\mathcal{D}^k_{S,v}$, let $\varphi(T_i)$ be the set:

- of vertices $g(T)$ of $\mathcal{C}^k_{S,v}$ such that T and T_i can be separated respectively from $S \cup \{v\}$ and from S by two parallel straight lines Δ and Δ' with same orientation and such that $T \subset \Delta^-$ and $T_i \subset \Delta'^-$,
- and of edges of $\mathcal{C}^k_{S,v}$ that connect such vertices.

Using basic properties of convex hulls, it is easy to see that $\varphi(T_i)$ is a connected polygonal line and that:

Proposition 3 *If $(g(T_1), ..., g(T_m))$ is the counterclockwise-ordered sequence of vertices of $\mathcal{D}^k_{S,v}$, then $\mathcal{C}^k_{S,v} = (\varphi(T_1), ..., \varphi(T_m))$.*

3 Triangulating $\mathbf{g^k(V)}$

We show now that the centroids of the k-sets of a convex inclusion chain of V are the vertices of a triangulation of $g^k(V)$ that has some common characteristics with the order-k Delaunay diagram of V. Recall that this diagram is dual to the order-k Voronoi diagram and that its vertices are the centroids of the k-element subsets of V that determine the order-k Voronoi regions of V. The order-k Delaunay diagram is then a triangulation of $g^k(V)$ whose every edge $g(T)g(T')$ is such that $|T \cap T'| = k-1$ [7, 11]. We show now that the centroids of the k-sets of any convex inclusion chain of V are also the vertices of such a triangulation. From Proposition 1, we already know that the edges of every k-set polygon fulfill the property. Moreover:

Proposition 4 *For every vertex $g(T_i)$ of $\mathcal{D}^k_{S,v}$ and for every vertex $g(T)$ of $\varphi(T_i)$, there exists $s \in T_i$ such that $T = (T_i \setminus \{s\}) \cup \{v\}$ and the segments $g(T_i)g(T)$ triangulate the polygon $\mathcal{P} = \overline{g^k(S \cup \{v\}) \setminus g^k(S)}$.*

Proof. (i) By definition, for every vertex $g(T)$ of $\varphi(T_i)$, there exist two parallel oriented straight lines Δ and Δ' such that $\Delta^- \cap S = T_i$ and $\Delta'^- \cap (S \cup \{v\}) = T$. Thus, there is a unique point s of S between Δ and Δ' and we have $T = (T_i \setminus \{s\}) \cup \{v\}$.

(ii) Now, it is not difficult to show that $g(T)$ can be separated from $g^k(S)$ by a straight line and thus that $g(T_i)g(T) \subset \mathcal{P}$. Moreover, from Proposition 3, two such segments can only intersect at their endpoints.

(iii) The boundary of $\mathcal{P}$ is composed of the edges of $\mathcal{D}^k_{S,v}$, of the edges of $\mathcal{C}^k_{S,v}$, and of the two other edges to create. From Proposition 3, for every edge $g(T)g(T')$ of $\mathcal{C}^k_{S,v}$, there exists a unique $i \in \{1, ..., m\}$ such that $g(T)g(T')$ is an edge of $\varphi(T_i)$. The triangle $g(T)g(T')g(T_i)$ splits then $\mathcal{P}$ into two simple polygons. In the same way, if $g(T_i)g(T_{i+1})$ is an edge of $\mathcal{D}^k_{S,v}$, $\varphi(T_i)$ and $\varphi(T_{i+1})$ have a common vertex $g(T)$ and the triangle $g(T)g(T_i)g(T_{i+1})$ splits $\mathcal{P}$ into two simple polygons. By induction, $\mathcal{P}$ can thus be triangulated by such triangles (see Figure 3). □

Now, if $\mathcal{V} = (v_1, v_2, ..., v_n)$ is a convex inclusion chain of V, by applying Proposition 4 successively to the subsets $S_k = \{v_1, ..., v_k\}, ..., S_n = \{v_1, ..., v_n\}$, we get:

Theorem 5 *The centroids of the k-sets of $\mathcal{V}$ are the vertices of a triangulation of $g^k(V)$ whose every edge $g(T)g(T')$ is such that $|T \cap T'| = k-1$.*

The triangulation determined by this theorem is called the order-k triangulation of $\mathcal{V}$ and is denoted by $\mathcal{T}^k(\mathcal{V})$ (see Figure 4).

It is easy to see that, if the edges of a triangulation fulfill Theorem 5 then, for every triangle $g(T)g(T')g(T'')$ of this triangulation, either $|T \cap T' \cap$

$T''| = k-1$ (called a type-1 triangle), or $|T \cup T' \cup T''| = k+1$ (called a type-2 triangle).

4 Order-k triangulations and centroid triangulations

Lee has proposed an algorithm to construct the order-k Voronoi diagram by starting with the (order-1) Voronoi diagram and iteratively computing the order-i diagram from the order-$(i-1)$ diagram [7]. This algorithm can be dualized to construct iteratively the order-k Delaunay diagram starting with the (order-1) Delaunay diagram [11]. The method to construct the order-i Delaunay diagram from the order-$(i-1)$ diagram is the following:

Algorithm 1

- *For every type-1 triangle* $g(P \cup \{r\})g(P \cup \{s\})$ $g(P \cup \{t\})$ *of* $Del_{i-1}(V)$ *compute the triangle* $g(P \cup \{r,s\})g(P \cup \{r,t\})g(P \cup \{s,t\})$.
- *The set* τ *of these triangles is the set of type-2 triangles of* $Del_i(V)$.
- *The type-1 triangles of* $Del_i(V)$ *are obtained by computing the constrained (order-1) Delaunay triangulation of* $\overline{g^i(V) \setminus \tau}$.

In [8], Lyu and Snoeyink conjectured that, starting with any triangulation of the point set V and computing any constrained triangulation at every step, this algorithm constructs triangulations whose edges verify Theorem 5 (they proved the result for $k \leq 3$). The triangulations generated in this way are called centroid triangulations. Here we show that, for all k:

Theorem 6 *The order-k triangulation of any convex inclusion chain is a centroid triangulation.*

Proof. For every point set S we call (centroid) triangulation sequence of S, any sequence $(\mathcal{A}^1, ..., \mathcal{A}^{|S|})$ of centroid triangulations such that $\mathcal{A}^1$ is a triangulation of S and, for all $i \in \{2, ..., |S|\}$, $\mathcal{A}^i$ is obtained from $\mathcal{A}^{i-1}$ by the generalization of Algorithm 1. Note that $\mathcal{A}^1$ contains only type-1 triangles, $\mathcal{A}^{|S|-1}$ only type-2 triangles, and $\mathcal{A}^{|S|} = g^{|S|}(S)$ is reduced to the unique point $g(S)$.

Suppose by induction that, for every set S of $n-1$ points, for every convex inclusion chain $\mathcal{S}$ of S, and for every positive integer $k \leq n-1$, the order-k triangulation $\mathcal{T}^k(\mathcal{S})$ of $\mathcal{S}$ is a centroid triangulation of S and that $(\mathcal{T}^1(\mathcal{S}), ..., \mathcal{T}^{n-1}(\mathcal{S}))$ is a triangulation sequence of S. This is trivially true for $n-1 = 1$.

Let now v be a point not belonging to $conv(S)$, $V = S \cup \{v\}$, and $\mathcal{V} = (\mathcal{S}, v)$. When v is added to S, $\mathcal{C}^1_{S,v}$ is composed of the unique vertex v and $\mathcal{D}^1_{S,v}$ is composed of the edges and vertices of the boundary of $\mathcal{T}^1(\mathcal{S})$ visible from v. By connecting these vertices to v, we get $\mathcal{T}^1(\mathcal{V})$ which is the first element of a triangulation sequence of V. Assume now, as a second induction hypothesis, that for a positive integer $h \leq n-1$, $(\mathcal{T}^1(\mathcal{V}), ..., \mathcal{T}^{h-1}(\mathcal{V}))$ is an initial part of a triangulation sequence of V. $\mathcal{T}^h(\mathcal{V})$ verifies Theorem 5 and, from the proof of Proposition 4, $\overline{\mathcal{T}^h(\mathcal{V}) \setminus \mathcal{T}^h(\mathcal{S})}$ has two kinds of triangles: Triangles with one edge on $\mathcal{D}^h_{S,v}$ and the opposite vertex on $\mathcal{C}^h_{S,v}$, and triangles with one edge on $\mathcal{C}^h_{S,v}$ and the opposite vertex on $\mathcal{D}^h_{S,v}$ (see Figure 3). Using Proposition 4, it can be shown that these triangles are respectively of type 1 and 2. With the argument that $e_P(s,t)$ is an edge of $\mathcal{C}^h_{S,v}$ if and only if $e_{P\setminus\{v\}}(s,t)$ is an edge of $\mathcal{D}^{h-1}_{S,v}$, the type-2 triangles of $\overline{\mathcal{T}^h(\mathcal{V}) \setminus \mathcal{T}^h(\mathcal{S})}$ can be obtained from the type-1 triangles of $\overline{\mathcal{T}^{h-1}(\mathcal{V}) \setminus \mathcal{T}^{h-1}(\mathcal{S})}$ by the generalization of Algorithm 1. From the first induction hypothesis, $\mathcal{T}^h(\mathcal{V})$ can then also be obtained from $\mathcal{T}^{h-1}(\mathcal{V})$ by this algorithm and, from the second induction hypothesis, $(\mathcal{T}^1(\mathcal{V}), ..., \mathcal{T}^h(\mathcal{V}))$ is an initial part of a triangulation sequence of V, for all $h \in \{1, ..., n-1\}$. Since $\mathcal{T}^n(\mathcal{V})$ is reduced to the centroid $g^n(V)$, it follows that $(\mathcal{T}^1(\mathcal{V}), ..., \mathcal{T}^n(\mathcal{V}))$ is a triangulation sequence of V. □

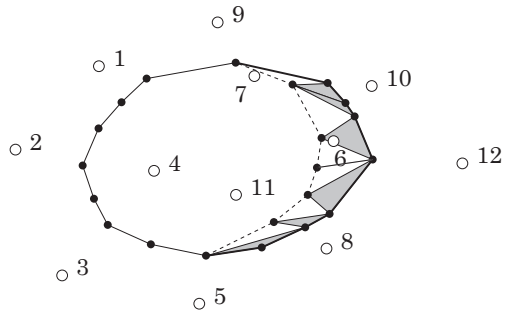


Figure 3: A triangulation of $\overline{g^4(S \cup \{12\}) \setminus g^4(S)}$, with $S = \{1, ..., 11\}$. The white triangles are of type 1 and the grey triangles are of type 2.

5 Construction of a centroid triangulation

As we know from [7] and [5], the order-k Delaunay diagram and the order-k triangulations of convex inclusion chains have both $2kn - n - k^2 + 1 - \sum_{j=1}^{k-1} \gamma^j(V)$ vertices (with $\gamma^j(V)$ the number of j-sets of V and $\sum_1^0 = 0$). Lyu and Snoeyink have conjectured that any centroid triangulation has $O(k(n-k))$ vertices. Thus, the generalization of Algorithm 1 constructs a centroid triangulation in at least $O(n \log n + k^2(n-k))$ time (at least $O(n \log n)$ for the order-1 triangulation and at least $O(k(n-k))$ for each of the $k-1$ other centroid triangulations). We show now that a particular centroid triangulation can be constructed in $O(n \log n + k(n-k) \log^2 k)$ time.

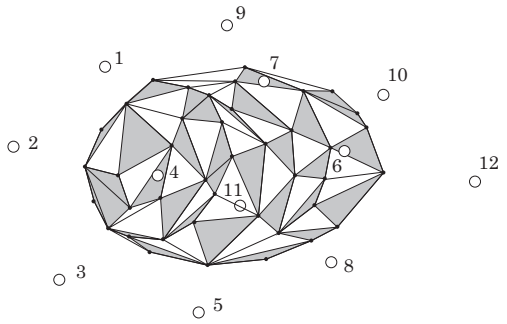


Figure 4: The order-4 triangulation of the convex inclusion chain $(2, 3, 1, 4, 5, 9, 11, 7, 8, 6, 10, 12)$.

To handle a centroid triangulation, we need to know its combinatorial structure, to maintain for every edge $g(P \cup \{s\})g(P \cup \{t\})$ a link to s and t, and to store the set T for exactly one vertex $g(T)$. Moreover, denoting by $|\mathcal{D}^k_{S,v}|$ and $|\mathcal{C}^k_{S,v}|$ the numbers of vertices of $\mathcal{D}^k_{S,v}$ and of $\mathcal{C}^k_{S,v}$, it has been shown in [5] that, if $g^k(S)$ and v are given, $\mathcal{D}^k_{S,v}$ and $\mathcal{C}^k_{S,v}$ can be found in $O(|\mathcal{D}^k_{S,v}| \log^2 k + |\mathcal{C}^k_{S,v}|)$ time, provided that one vertex $g(T_i)$ of $\mathcal{D}^k_{S,v}$ is known and that the convex hull of T_i is stored in a fully dynamic convex hull data structure (see [9]). Then we have:

Theorem 7 *V admits a centroid triangulation that can be constructed in $O(n \log n + k(n-k) \log^2 k)$ time.*

Proof. Let $\mathcal{V} = (v_1, ..., v_n)$ be a sequence of the points of V sorted by increasing x-coordinates. Obviously, $\mathcal{V}$ is a convex inclusion chain of V. For every subset $S_j = \{v_1, ..., v_j\}$ $(j \in \{k, ..., n-1\})$, it can be shown that $\mathcal{D}^k_{S_j, v_{j+1}}$ contains a vertex of $g^k(S_j)$ with maximal x-coordinate, i.e., the centroid of k points of S_j with maximal x-coordinates. If we maintain the dynamic convex hull of these k points and use the triangulation method of the proof of Proposition 4, given $g^k(S_j)$, a triangulation of $\overline{g^k(S_{j+1}) \setminus g^k(S_j)}$ can be constructed in $O(|\mathcal{D}^k_{S_j, v_{j+1}}| \log^2 k + |\mathcal{C}^k_{S_j, v_{j+1}}|)$ time.

Starting with $g^k(S_k) = g(S_k)$ and applying this triangulation method for all $j \in \{k, ..., n-1\}$, we get an order-k triangulation of $\mathcal{V}$ in $O(\sum_{j=k}^{n-1} |\mathcal{D}^k_{S_j, v_{j+1}}| \log^2 k + \sum_{j=k}^{n-1} |\mathcal{C}^k_{S_j, v_{j+1}}|)$ time. Now, $\sum_{j=k}^{n-1} |\mathcal{C}^k_{S_j, v_{j+1}}| + 1$ is the total number of vertices of the order-k triangulation of $\mathcal{V}$ and it is easy to see that $\sum_{j=k}^{n-1} |\mathcal{D}^k_{S_j, v_{j+1}}| < \sum_{j=k}^{n-1} |\mathcal{C}^k_{S_j, v_{j+1}}|$. Since the total number of vertices of the order-k triangulation of $\mathcal{V}$ is $O(k(n-k))$, this triangulation can be constructed in $O(k(n-k) \log^2 k)$ time, after having sorted V. □

6 Conclusion

In this paper, we have shown that the family of centroid triangulations of a planar point set, which is known to contain the order k-Delaunay diagram, also contains the family of order-k triangulations of the convex inclusion chains of the point set.

Now, if we were able to show that all the centroid triangulations have the same number of vertices, this would completely explain why the number of k-sets of a convex inclusion chain is equal to the number of regions of the order-k Voronoi diagram. To achieve this goal, we will probably need to find a geometric characterization of centroid triangulations.

References

[1] A. Andrzejak and K. Fukuda. Optimization over k-set polytopes and efficient k-set enumeration. In *Proc. 6th Workshop Algorithms Data Struct.*, volume 1663 of *Lecture Notes Comput. Sci.*, pages 1–12. Springer-Verlag, 1999.

[2] A. Andrzejak and E. Welzl. In between k-sets, j-facets, and i-faces: (i, j)-partitions. *Discrete Comput. Geom.*, 29:105–131, 2003.

[3] F. Aurenhammer and O. Schwarzkopf. A simple online randomized incremental algorithm for computing higher order Voronoi diagrams. *Internat. J. Comput. Geom. Appl.*, 2:363–381, 1992.

[4] T. K. Dey. Improved bounds on planar k-sets and related problems. *Discrete Comput. Geom.*, 19:373–382, 1998.

[5] W. El Oraiby and D. Schmitt. k-sets of convex inclusion chains of planar point sets. In *Proc. 31st Intern. Symp. Math. Found. Comput. Sci.*, volume 4162 of *LNCS*, pages 339–350. Springer-Verlag, 2006.

[6] J. Gudmundsson, M. Hammar, and M. J. van Kreveld. Higher order Delaunay triangulations. *Comput. Geom.*, 23(1):85–98, 2002.

[7] D. T. Lee. On k-nearest neighbor Voronoi diagrams in the plane. *IEEE Trans. Comput.*, C-31:478–487, 1982.

[8] Y. Liu and J. Snoeyink. Quadratic and cubic b-splines by generalizing higher-order Voronoi diagrams. In *Symposium on Computational Geometry*, pages 150–157, 2007.

[9] M. H. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. *J. Comput. Syst. Sci.*, 23:166–204, 1981.

[10] G. W. Peck. On k-sets in the plane. *Discrete Math.*, 56:73–74, 1985.

[11] D. Schmitt and J.-C. Spehner. On Delaunay and Voronoi diagrams of order k in the plane. In *Proc. 3rd Canad. Conf. Comput. Geom.*, pages 29–32, 1991.

[12] G. Tóth. Point sets with many k-sets. *Discrete Comput. Geom.*, 26(2):187–194, 2001.

Constructing the Segment Delaunay Triangulation by Flip

Mathieu Brévilliers* Nicolas Chevallier* Dominique Schmitt*

Abstract

Using locally convex functions, we show that the dual of the segment Voronoi diagram in the plane can be computed by a flip algorithm.

1 Introduction

The flip algorithm is a classical method to construct the Delaunay triangulation of a set of points in the plane, starting with any given triangulation [6]. In recent years, the method has been extended to generalized triangulations of point sets such as pseudo-triangulations or pre-triangulations [2], [3], [1], ...

In this paper, we propose a flip algorithm to construct the dual of the Voronoi diagram of a set of segments in the plane. This diagram, called segment Delaunay triangulation, has been introduced by Chew and Kedem [5]. In [4], we have already defined a family of diagrams containing the segment Delaunay triangulation: the segment triangulations (see Figure 1). The faces of such a triangulation form a maximal set of disjoint triangles resting on three distinct segments.

A classical method to study flip algorithms consists in lifting the triangulations to three-dimensional space. The problem here is that lifting has to be performed on non-convex regions in the plane. As in [2] and [3], we overcome this problem with the help of locally convex functions.

Another difficulty comes out of the fact that there are infinitely many segment triangulations of a given segment set. Thus, we give a flip algorithm that constructs, in a finite number of steps, a segment triangulation that has the same topology as the segment Delaunay triangulation.

2 Segment triangulations

In this section, we recall the main results about segment triangulations given in [4].

Throughout this paper, S is a finite set of $n \geq 2$ disjoint closed segments in the plane, which we call sites. A closed segment may possibly be reduced to a single point. We say that a circle is tangent to a site s if s meets the circle but not its interior. The sites of S are supposed to be in general position, that is, we suppose that no three segment endpoints are collinear and that no circle is tangent to four sites.

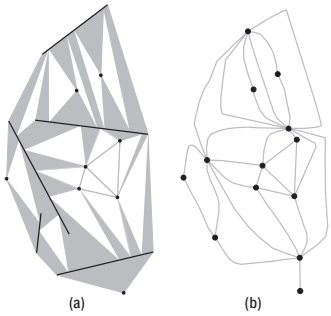


Figure 1: A segment triangulation (a) (the sites appear in black, the faces in white, and the edges in grey) and its topology (b).

Definition 1 *A segment triangulation $\mathcal{T}$ of S is a partition of the convex hull $conv(S)$ of S in disjoint sites, edges, and faces such that:*
1. Every face of $\mathcal{T}$ is an open triangle whose vertices belong to three distinct sites of S and whose open edges do not intersect S,
2. No face can be added without intersecting another one,
3. The edges of $\mathcal{T}$ are the (possibly two-dimensional) connected components of $conv(S) \setminus (F \cup S)$, where F is the set of faces of $\mathcal{T}$.

In the following, the word "triangle" will only be used for faces and never for edges, even if they have the shape of a triangle.

An edge of such a triangulation is adjacent to exactly two sites (see Figure 1). Moreover, the set of sites and edges defines a planar graph and thus a combinatorial map which represents the topology of the segment triangulation. The number of faces of a segment triangulation of S depends only on S and is linear with the number of sites of S.

Definition 2 *A segment triangulation of S is Delaunay if the circumcircle of each face does not contain any point of S in its interior.*

The segment Delaunay triangulation of S always exists. Moreover, since S is in general position, it is unique and dual to the segment Voronoi diagram of S. Note that the geometry of the segment Delaunay triangulation is easy to compute once its topology is

*Laboratoire MIA, Université de Haute-Alsace, Mulhouse, France {Mathieu.Brevilliers, Nicolas.Chevallier, Dominique.Schmitt}@uha.fr

known. Indeed, it suffices to put every triangle t in tangency position on the three sites on which it rests, i.e., its circumcircle is tangent to these three sites and meets them in the same order as t.

As for point sets, a segment Delaunay triangulation can be recognized with local tests using edge legality. An edge of a segment triangulation is said to be (topologically) legal if the triangles adjacent to the edge, moved to their tangency positions, are Delaunay with respect to the sites adjacent to the triangles and if they retain locally the original topology. Hence:

Theorem 1 *A segment triangulation of S whose all edges are legal has the same topology as the Delaunay one.*

It is easy to see that the legality of an edge can be checked in constant time. Thus, there is a linear time algorithm that checks whether a given segment triangulation has the same topology as the segment Delaunay triangulation.

In this paper, we shall need to constrain the segment triangulations in some subsets of the convex hull of S. Thus, we extend slightly the above results.

Definition 3 *A subset U of $conv(S)$ is S-polygonal if U is closed and if the boundary of U is a finite union of disjoint segments of two kinds:*
– closed segments included in S,
– open segments $]p,q[$ such that $S \cap [p,q] = \{p,q\}$.

Now, the definition of segment triangulations extends to an S-polygonal subset U of $conv(S)$ by replacing, in Definition 1, $conv(S)$ by U and S by $U \cap S$. Here again we can show that the number of faces of a segment triangulation of U depends only on the couple (U, S).

Definition 4 *A segment triangulation $\mathcal{T}$ of U is Delaunay if the interior of the circumcircle of each triangle t of $\mathcal{T}$ contains no point of S that is visible from an interior point of t, i.e., the open segment connecting these two points is not included in $U \setminus S$.*

Theorem 5 of section 4, shows that a segment Delaunay triangulation of U always exists. However, it is not necessarily unique since four connected components of $U \cap S$ may be cocircular even if S is in general position.

3 Description of the flip algorithm

The algorithm starts with a segment triangulation of S. The edges of the triangulation are stored in a queue. The edge e at the head of the queue is popped and a Delaunay triangulation of the S-polygonal subset P, union of e and of its adjacent triangles, is constructed (P is called the input polygon of e; see Figure 2). This gives rise to a new segment triangulation.

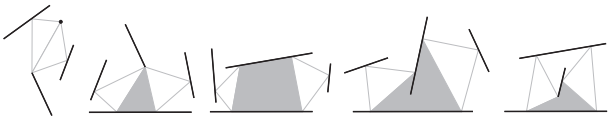


Figure 2: Input polygons of some edges.

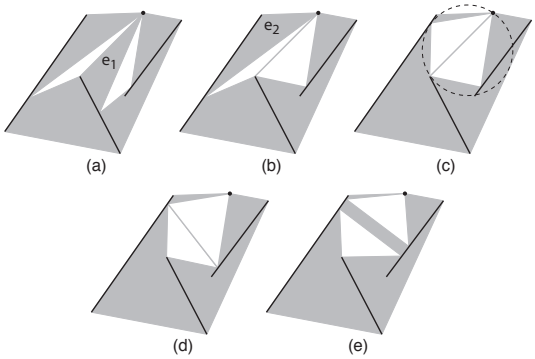


Figure 3: The flip algorithm transforms the given segment triangulation (a) in a segment triangulation (d) that has the same topology as the Delaunay one (e). The edge e_1 of (a) is treated but remains illegal in (b) because it cannot be flipped. The legal edge e_2 has to be processed before the flip of e_1.

The edge replacing e is pushed at the tail of the queue. Beside this queue, a list of illegal edges is maintained. The algorithm ends when all edges are legal.

Studying the different cases, we can show that a Delaunay triangulation of P can be computed in constant time. If this triangulation admits two triangles and if the edge between them does not connect the same two sites as the edge used to determine P, then the edge is said to be flipped.

Even if the algorithm looks very close to the classical flip algorithm, there are important differences in their convergences. In case of segment sets:
– some illegal edges cannot be flipped (see Figure 3),
– a new constructed edge is not necessarily legal,
– a removed topological edge can reappear (Figure 4).

This shows that neither the legality of the edges nor the flip count suffices to prove the convergence of the algorithm. Another way to prove the convergence of the point set flip algorithm to the Delaunay triangulation, is to lift the point set on the three-dimensional paraboloid $z = x^2 + y^2$. It is well known that the downward projection of the lower convex hull of the lifting is the Delaunay triangulation of the point set. Conversely, every other triangulation lifts to a non convex polyhedral surface above the lower convex hull. Now, it is enough to notice that an edge flip brings down the polyhedral surface.

The lower convex hull of a set S of segments lifted on the paraboloid, also projects downward onto the segment Delaunay triangulation of S (see Theorem 5). The main difficulty is to lift the other segment

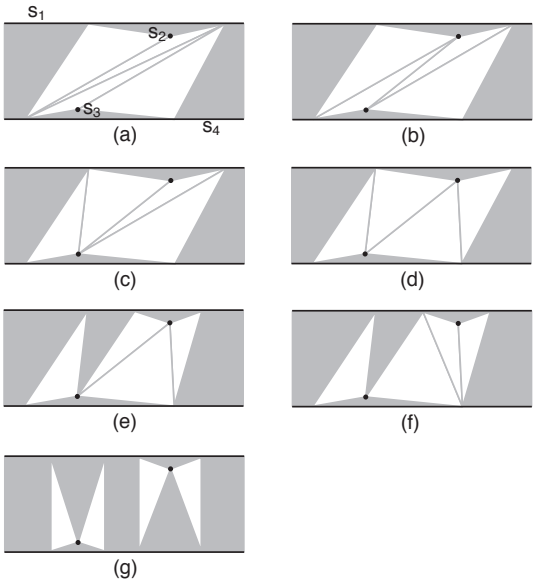


Figure 4: The edge flipped between (a) and (b) remains illegal. The edge connecting s_2 and s_4 in (b) disappears in (c) and reappears in (d).

triangulations and especially their non convex edges. To this aim we use locally convex functions.

4 Locally convex functions and segment triangulations

Recall that a real-valued function ϕ on the line segment s is convex if $\phi(tx+(1-t)y) \leq t\phi(x)+(1-t)\phi(y)$, for all t in $[0,1]$ and all $x, y \in s$. More generally, if V is a subset of $\mathbf{R}^2$ and $\phi : V \to \mathbf{R}$ is a function, we say that ϕ is locally convex if the restriction of ϕ to each segment included in V is convex.

We define now the lower convex hull of a function, which we shall use instead of the usual lower convex hull of a subset in $\mathbf{R}^3$.

Definition 5 *Given a real-valued function f defined on $V \cap S$, the lower convex hull of f on (V, S) is $f_{V,S} = \sup\{\phi : V \to \mathbf{R} : \phi \in L(V),\ \forall x \in V \cap S,\ \phi(x) \leq f(x)\}$ where $L(V)$ is the set of functions $\phi : V \to \mathbf{R}$ that are locally convex on V.*

In the following, U denotes an S-polygonal subset of $conv(S)$ and the above definition will be used with the function $f : \mathbf{R}^2 \to \mathbf{R}$ defined by $f(x, y) = x^2+y^2$. The convexity of f implies that $f_{U,S} = f$ on $U \cap S$. It can also be proven that $f_{U,S}$ is continuous.

The main aim of this section is to explain that the function $f_{U,S}$ determines a segment Delaunay triangulation of U. Next theorem gives information about the value of the function $f_{U,S}$ at a point p. It begins by the simplest case where U is convex. Then it shows how to reduce the general case to the convex case. For every point p in $U \setminus S$, denote S_p the closure of the set of points in S visible from p and V_p its convex hull (in general, V_p is not contained in U). The theorem asserts that $f_{U,S}(p)$ depends only on the lower convex hull of f on (V_p, S_p).

Theorem 2
1. If U is convex, then every point of $U \setminus S$ belongs to a closed convex subset C of U whose extremal points are in S and such that $f_{U,S}$ is affine on C.
2. In case of a (non convex) S-polygonal subset U, let p be a point of $U \setminus S$. If C is a closed convex subset of V_p, containing p, whose extremal points are in S_p, and such that f_{V_p,S_p} is affine on C, then C is included in U and $f_{U,S} = f_{V_p,S_p}$ on C.

The next step consists in showing that U can be partitioned into maximal convex subsets where the function $f_{U,S}$ is affine.

Theorem 3 *Every point p in $U \setminus S$ belongs to a convex subset C_p that is maximal for the inclusion among the relatively open convex subsets of U where $f_{U,S}$ is affine. Moreover, the extremal points of $\overline{C_p}$ are in S and, if q is another point of $U \setminus S$, either $C_p \cap C_q = \emptyset$, or $C_p = C_q$.*

The last statement of Theorem 3 means that the subsets C_p form a partition of $U \setminus S$. Now we have to establish that the two-dimensional convex subsets among the C_p are the faces of a segment triangulation.

Theorem 4 *By decomposing the two-dimensional $(C_p)_{p \in U \setminus S}$ into triangles we get the faces of a segment triangulation $\mathcal{T}$ of U, which we call a triangulation induced by $f_{U,S}$.*

Suppose now that $U = conv(S)$ and let t be a triangle of $\mathcal{T}$ and h the affine function that is equal to $f_{U,S}$ on t. The graph of h is a plane and its intersection with the graph of f is an ellipse whose downward projection is the circumcircle of t. Since U is convex, the function $f_{U,S}$ is convex. Therefore, $h \leq f_{U,S}$ on U. It follows that $h \leq f$ on $S \cap U$. We deduce that the circumcircle of t does not contain any point of S in its interior. By definition of a face of a segment Delaunay triangulation, we conclude that:

Theorem 5
1. If $U = conv(S)$, the segment triangulation induced by $f_{U,S}$ is the segment Delaunay triangulation of S.
2. For any S-polygonal subset U, a segment triangulation of U is induced by $f_{U,S}$ if and only if it is Delaunay.

Using locally convex functions, we are able to lift any segment triangulation in the following way:

Definition 6 *Let $\mathcal{T}$ be a segment triangulation of U. The function $f_{U,S,\mathcal{T}} : U \to \mathbf{R}$ is equal to f on S, to $f_{\overline{e},S}$ on any edge e of $\mathcal{T}$, and to $f_{t,S}$ on the interior of any triangle t of $\mathcal{T}$.*

The lifting of $\mathcal{T}$ to $\mathbf{R}^3$ is the graph of the function $f_{U,S,\mathcal{T}}$. Using the previous results we have then:

Theorem 6
1. If $\mathcal{T}$ is a segment triangulation of U, then $f_{U,S} \leq f_{U,S,\mathcal{T}}$. Moreover $f_{U,S} = f_{U,S,\mathcal{T}}$ if and only if $\mathcal{T}$ is induced by $f_{U,S}$.
2. If $U = conv(S)$, then $\mathcal{T}$ is the segment Delaunay triangulation of S if and only if $f_{U,S} = f_{U,S,\mathcal{T}}$.

5 Convergence of the flip algorithm

In case of point set triangulations, it is well known that a flip increases the smallest angle of the triangles. A weaker result holds for segment triangulations.

Given a segment triangulation $\mathcal{T}$ of U, let the slope of $\mathcal{T}$ be:

$$\sigma(\mathcal{T}) = \sup\{\tfrac{f_{U,S,\mathcal{T}}(p)-f_{U,S,\mathcal{T}}(q)}{|p-q|} : p \in U \setminus S,\ q \in U \cap S,\ [p,q] \subset U\}$$

Denoting by $\theta(\mathcal{T})$ the minimal angle of the triangles of $\mathcal{T}$, we get then:

Proposition 7 *There exists a positive constant c depending only on f, S, and U such that, for every segment triangulation $\mathcal{T}$ of U, $\theta(\mathcal{T}) \geq c/(\max(1, \sigma(\mathcal{T})))$.*

It is not difficult to prove that $\sigma(\mathcal{T}) < +\infty$ and, if $\mathcal{T}'$ is a segment triangulation of U such that $f_{U,S,\mathcal{T}} \leq f_{U,S,\mathcal{T}'}$, then $\sigma(\mathcal{T}) \leq \sigma(\mathcal{T}')$.

Consider now our algorithm: It starts with a segment triangulation $\mathcal{T}_0$ of $conv(S)$ and computes a sequence $\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_n, ...$ of triangulations.

Theorem 8 *The sequence $(f_n = f_{conv(S),S,\mathcal{T}_n})_{n\in\mathbf{N}}$ decreases to $f_{conv(S),S}$ as n goes to infinity.*

Proof. At every stage n, we compute a Delaunay triangulation of the input polygon P_n of the edge at the head of the queue. Applying Theorem 6 to the S-polygonal subset U composed of P_n and of all the edges of $\mathcal{T}_n$ adjacent to P_n, we get that $f_{n+1} \leq f_n$ on U which implies that $f_{n+1} \leq f_n$ on $conv(S)$.

It follows that the sequence of functions $(f_n)_{n\in\mathbf{N}}$ decreases to a function $g : conv(S) \to \mathbf{R}$. The only thing to show is that g is locally convex, i.e., g is convex on any open segment $]p_0, p_1[$ included in the interior of $conv(S)\backslash S$. Since the angles of the triangles generated by the algorithm are not too sharp, it can be shown that, for every point p of $]p_0, p_1[$, there exists $\varepsilon > 0$ such that the neighbourhood $I_{p,\varepsilon}$ of p of lenght ε in $]p_0, p_1[$ is included either in a triangle of $\mathcal{T}_n$ or in the input polygon P_n treated at stage n, for infinitely many integers n. Thus, for these integers n, either f_n or $f_{P_n,S}$ is convex on $I_{p,\varepsilon}$, and since $f_{n+1} \leq f_{P_n,S} \leq f_n$ on P_n, the function g is a limit of a sequence of convex functions on $I_{p,\varepsilon}$. □

Now, note that the set of topologies of all the segment triangulations of S is finite. We can also show that the only topology that appears infinitely many times in the sequence $(\mathcal{T}_n)_{n\in\mathbf{N}}$, is the topology of the segment Delaunay triangulation. Thus:

Corollary 9 *There exists an integer N such that, for all integers $n \geq N$, the triangulation $\mathcal{T}_n$ has the same topology as the segment Delaunay triangulation of $conv(S)$.*

6 Conclusion

The aim of this paper was to show that the dual of the segment Voronoi diagram can be constructed by a flip algorithm in a finite number of steps. The remaining computational problems concern the implementation of the algorithm: robustness, time complexity,... The algorithm has also to be compared with standard methods for computing segment Voronoi diagrams.

From the theoretical point of view, the fact that the angles of the triangles that appear during the algorithm cannot be too sharp, makes us believe that the segment Delaunay triangulation should have some optimality properties.

At last, possible extensions of segment triangulations should be mentioned: Extension to three-dimensional space, to more general sites, to more general distance functions, ...

References

[1] O. Aichholzer, F. Aurenhammer, and T. Hackl. Pretriangulations and liftable complexes. In *Proc. 22th Annu. ACM Sympos. Comput. Geom.*, pages 282–291, 2006.

[2] O. Aichholzer, F. Aurenhammer, H. Krasser, and P. Brass. Pseudotriangulations from surfaces and a novel type of edge flip. *SIAM J. Comput.*, 32(6):1621–1653, 2003.

[3] F. Aurenhammer and H. Krasser. Pseudo-simplicial complexes from maximal locally convex functions. *Discrete and Computational Geometry*, 35(2):201–221, 2006.

[4] M. Brévilliers, N. Chevallier, and D. Schmitt. Triangulations of line segment sets in the plane. In *FSTTCS*, pages 388–399, 2007.

[5] L. P. Chew and K. Kedem. Placing the largest similar copy of a convex polygon among polygonal obstacles. In *Proc. 5th Annu. ACM Sympos. Comput. Geom.*, pages 167–174, 1989.

[6] C. L. Lawson. Software for C^1 surface interpolation. In J. R. Rice, editor, *Math. Software III*, pages 161–194. Academic Press, New York, NY, 1977.

Intersection Graphs of Pseudosegments and Chordal Graphs: An Application of Ramsey Theory

Cornelia Dangelmayr* Stefan Felsner† William T. Trotter‡

Abstract

We investigate which chordal graphs have a representation as intersection graph of pseudosegments. In previous work we have shown that all chordal graphs which can be represented as vertex intersection graph of subpaths in a tree are pseudosegment intersection graphs. It was also shown that not all chordal graphs are intersection graphs of pseudosegments. Studying the limits of representability we now investigate chordal graphs that can be represented as vertex intersection graphs of substars of a star, where the substars are restricted to have degree at most three. Using a Ramsey argument we show that there are graphs in this class that are not representable as intersection graph of pseudosegments.

1 Introduction

A family of pseudosegments is understood to be a set of Jordan arcs in the Euclidean plane that are pairwise either disjoint or intersect at a single crossing point. A family of pseudosegments represents a graph G, the vertices of G are the Jordan arcs and two vertices are adjacent if and only if the corresponding arcs intersect. A graph represented by a family of pseudosegments is a *pseudosegment intersection graph*, for short a PSI-graph.

PSI-graphs are sandwiched between the larger class of string-graphs (intersection graphs of Jordan arcs without condition on their intersection behavior) and of segment-graphs (intersection graphs of straight line segments). In one of the first papers on this subject Ehrlich et al. [4] proved that all planar graphs are string-graphs. In fact this also follows from Koebe's coin graph theorem. Scheinerman in his thesis [7] conjectured that planar graphs are segment graphs. See [3] and [1] for contributions to this problem. In general the recognition of PSI-graphs is NP-complete [6].

There are some classes of graphs where segment representation, hence, as well PSI representations, are trivial (e.g. permutation graphs and circle graphs) or very easy to find (e.g. interval graphs). A large superclass of interval graphs is the class of chordal graphs. In [5] Gavril characterized chordal graphs as the vertex intersection graphs of subtrees of a tree. In [2] we have shown that path graphs, i.e. vertex intersection graphs of paths in a tree, have a PSI-representation. If we allow the subtrees to be stars (of large degree), or (large) caterpillars of maximum degree three, there need not exist a PSI-representation [2]. We define a graph K_n^3 that is not in PSI for $n \geq 33$. The vertex set of K_n^3 is partitioned as $V = V_C \cup V_I$ such that $V_C = [n]$ induces a clique and $V_I = \binom{[n]}{3}$ is an independent set. The edges between V_C and V_I represent membership, i.e, $\{i, j, k\} \in V_I$ is connected to the vertices i, j and k from V_C. K_n^3 can be represented as vertex intersection graph of subtrees of a star S with $\binom{n}{3}$ leaves. The leaves correspond to the elements of V_I. Each $v \in V_I$ is represented by a trivial tree with only one node. A vertex i of the complete graph on V_C is represented by the star S_i connecting to all leaves of triples containing i. This shows that K_n^3 is chordal. The central node of the star S_i has high degree. If we take a path of $\binom{n}{3}$ nodes and attach a leaf-node to each node of the path we obtain a tree T of maximum degree three such that the graph K_n^3 can be represented as vertex intersection graph of subtrees of T. Actually the tree T and its subtrees are caterpillars of maximum degree three.

These remarks show that the positive and the negative result of [2] leave little room for questions. Open remains the PSI-representation of vertex intersection graphs of substars with bounded degree of a star. Let S_n be the chordal graph whose vertices are represented by all substars with three leaves and all leaves on a star with n leaves. The following conjecture was stated in [2]:

Conjecture: *For n large enough S_n has no PSI-representation.*

2 Proof of the Conjecture

Our proof makes use of a Ramsey argument, hence, we need a really large n for the result. Suppose there is a PSI-representation of S_n. That is, there is a set $\mathcal{D}$ of n pairwise disjoint pseudosegments representing the substars that are leaves of S and a set Γ of pairwise

*Freie Universität Berlin, Mathematisches Institut, Arnimallee 3, 14195 Berlin, Germany, dangel@math.fu-berlin.de

†Technische Universität Berlin, Institut für Mathematik, MA 6-1, Strasse des 17. Juni 136, 10623 Berlin, Germany, felsner@math.tu-berlin.de

‡School of Mathematics, Georgia Institute of Technology, Atlanta, Georgia 30332-0160, USA, trotter@gatech.edu

intersecting pseudosegments representing the substars of S that have three leaves. We obtain a contradiction in the following two steps.

1. We apply Ramsey Theory to prove the existence of a 'regular' PSI-representation of a subgraph S_m of S_n in a PSI-representation of S_n if $n \in \mathbb{N}$ is large enough.
2. By a geometric argument we show that the existence of a regular PSI-representation of a graph S_m for $m \geq 6$ requires multiple intersections between pairs of pseudosegments of Γ. This is the contradiction as pairs of pseudosegments intersect at most once.

To simplify the picture we first transform the plane such that the pseudosegments of $\mathcal{D}$ get vertical segments of unit length which touch the X-axis with their lower endpoints at positions $1, 2, \ldots, n$. Let $p_i \in \mathcal{D}$ be the pseudosegment containing the point $(i, 0)$. To elements of $\mathcal{D}$ we refer as *sticks*.

With every ordered triple (i, j, k), $1 \leq i < j < k \leq n$, there is a 3-segment γ_{ijk} intersecting p_i, p_j and p_k. Let ϕ_{ijk} denote the middle of the three sticks intersected by γ_{ijk}. We partition the ordered triples (i, j, k) into three classes depending of the position of ϕ_{ijk} in the list (p_i, p_j, p_k). If $\phi_{ijk} = p_i$, i.e., the middle intersection of γ_{ijk} is on the stick left of the other two, we assign (i, j, k) to class $[L]$. Analogously assign (i, j, k) to class $[M]$ if ϕ_{ijk} is the middle, and to $[R]$ if ϕ_{ijk} is the right most stick of (p_i, p_j, p_k). This will be one part of our classification.

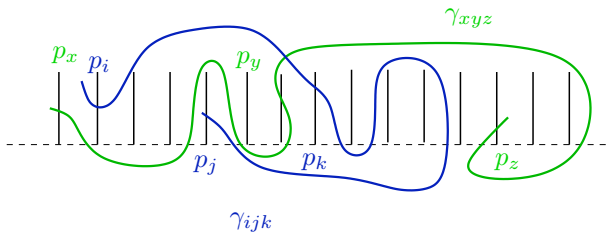


Figure 1: Two 3-segments γ_{ijk} and γ_{xyz}. Note that $\phi_{ijk} = p_k$ and $\phi_{xyz} = p_y$, hence, $\gamma_{ijk} \in [R]$ and $\gamma_{xyz} \in [M]$.

The other part is obtained by cutting γ_{ijk} at the intersection points with p_i, p_j and p_k. This yields two arcs γ^1_{ijk} and γ^2_{ijk} each connecting two of the sticks of $\mathcal{D}$ and up to two ends. The ends are of no further interest. For the arcs we adopt the convention that γ^1_{ijk} connects ϕ_{ijk} to the stick further left and γ^2_{ijk} connects ϕ_{ijk} to the stick further right. Let $\vec{r}_x$ be a vertical ray downwards starting at $(x, 0)$, i.e., the ray pointing down from the lower end of stick p_x. Let $I^s_x(ijk)$ be the number of intersections of ray $\vec{r}_x$ with γ^s_{ijk} and let $J^s_x(ijk)$ be the parity of $I^s_x(ijk)$, i.e., $J^s_x(ijk) = I^s_x(ijk) \pmod 2$.

Given an ordered 7-tuple (a, i, b, j, c, k, d), we call γ_{ijk} the induced 3-segment and let $T^s_x = J^s_x(ijk)$. The *pattern* of the tuple is the binary 8-tuple

$$(T^1_a, T^1_b, T^1_c, T^1_d, T^2_a, T^2_b, T^2_c, T^2_d).$$

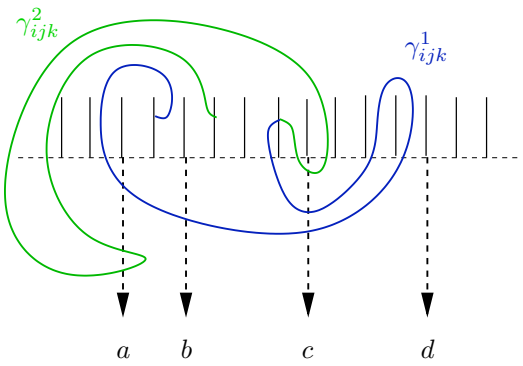


Figure 2: A 3-segment $\gamma_{ijk} \in [R]$ whose pattern with respect to a, b, c and d is $(1, 1, 0, 0, 0, 0, 1, 0)$, the third entry T^1_c is 0 because $I^1_c(ijk) = 2 \equiv 0 \pmod 2$.

The *color* of a 7-tuple (a, i, b, j, c, k, d) is the pair consisting of the class of the induced 3-segment and the pattern. The 7-tuples are thus colored with the 768 colors from the set $[3] \times \mathbf{2}^8$. Ordered 7-tuples and 7-element subsets of $[n]$ are essentially the same. Therefore we can apply the following Ramsey Theorem with parameters $768, 7, 6$.

Theorem: *For every choice of numbers r, p, k there exists a number n such that whenever X is an n-element set and c is a coloring of the system of all p-element subsets of X using r colors, then there is an k-element subset $Y \subseteq X$ such that all the p-subsets in $\binom{Y}{p}$ have the same color.*

This leaves us with a subset Y of sticks such that all 3-segments connecting three of them are of the same class and all 7-tuples on Y have the same pattern $T = (T^1_1, T^1_2, T^1_3, T^1_4, T^2_1, T^2_2, T^2_3, T^2_4)$. We will show that there have to be two 3-segments γ_{ijk} and γ_{xyz} such that γ^1_{ijk} and γ^1_{xyz} intersect, and so do γ^2_{ijk} and γ^2_{xyz}, hence, γ_{ijk} and γ_{xyz} intersect at least twice which is the contradiction we are striving for. To do so we associate with an arc γ_{ab} connecting sticks p_a and p_b a closed curve $\breve{\gamma}_{ab}$ as follows: At the intersection of γ_{ab} with either of the sticks we append long vertical segments and connect the lower endpoints of these two segments horizontally. The union of the three connecting segments will be called the *bow* β_{ab} of the curve $\breve{\gamma}_{ab}$. If this construction is applied to several arcs we assume that the vertical segments of the bows are long enough as to avoid any intersection between the arcs and the horizontal part of the bows and any pair of horizontal segments.

Set $X(\gamma, \gamma')$ as the number of crossings of curves γ and γ'. Then we can count the crossings of any pair of curves $\breve{\gamma}_{ab}$ and $\breve{\gamma}_{xy}$ as:

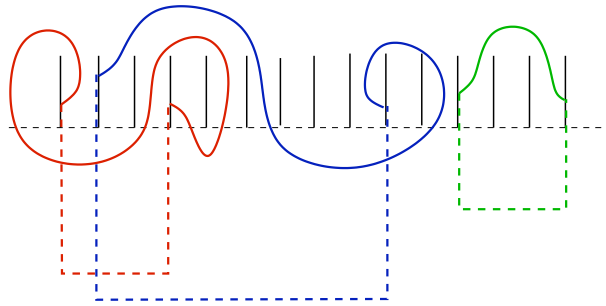


Figure 3: Three arcs γ and the corresponding closed curves $\breve{\gamma}$.

$$X(\breve{\gamma}_{ab}, \breve{\gamma}_{xy}) = X(\gamma_{ab}, \gamma_{xy}) + X(\gamma_{ab}, \beta_{xy}) + X(\beta_{ab}, \gamma_{xy}) + X(\beta_{ab}, \beta_{xy})$$

Fact 1 *If γ and γ' are closed curves, then $X(\gamma, \gamma') \equiv 0 \pmod 2$.*

Fact 1 is a consequence of the Jordan Curve Theorem. From this we obtain

Fact 2

$$X(\gamma_{ab}, \gamma_{xy}) \equiv X(\gamma_{ab}, \beta_{xy}) + X(\beta_{ab}, \gamma_{xy}) + X(\beta_{ab}, \beta_{xy}) \pmod 2.$$

Consider a linear order on $\{i, j, k, x, y, z\}$. The parity of the intersections between arcs of type γ^s_{ijk} and bows β_{ab} with $a, b \in \{x, y, z\}$ can be read off the pattern T. This can be used, e.g. to prove the following Lemmas:

Lemma 1 *If $T^1_1 = T^1_3$ and $i < x < j < y < k$, then there is an intersection between the arcs γ_{ij} and γ_{xy}.*

Proof. We evaluate the right side of the congruence given in Fact 2.

$X(\gamma_{ij}, \beta_{xy})$ is the number of intersections of arc γ_{ij} with the bow connecting p_x and p_y. These intersections happen on the vertical part, hence on the rays $\vec{r}_x$ and $\vec{r}_y$. The parity of these intersections can be read from the pattern. The position of x between i and j implies $T^1_x = T^1_2$ and the position of y between j and k implies $T^1_y = T^1_3$. Hence, $X(\gamma_{ij}, \beta_{xy}) \equiv T^1_2 + T^1_3 \pmod 2$.

From the positions of i left of x and of j between x and y we conclude that $X(\beta_{ij}, \gamma_{xy}) \equiv T^1_1 + T^1_2 \pmod 2$.

Since the pairs ij and xy interleave the two bows are intersecting, i.e., $X(\beta_{ij}, \beta_{xy}) = 1$.

Together this yields $X(\gamma_{ij}, \gamma_{xy}) \equiv T^1_2 + T^1_3 + T^1_1 + T^1_2 + 1 \pmod 2$. With $T^1_1 = T^1_3$ we see that $X(\gamma_{ij}, \gamma_{xy})$ is odd, hence, there is at least one intersection between the arcs. □

Lemma 2 *If $T^1_1 \neq T^1_3$ and $x < i < j < y < k$, then there is an intersection between the arcs γ_{ij} and γ_{xy}.*

Now consider the case where the class of all 3-segments is $[M]$. In addition to the arcs γ_{ij} and γ_{xy} we have the arcs $\gamma_{jk} = \gamma^2_{ijk}$, and $\gamma_{yz} = \gamma^2_{xyz}$. The following two lemmas are counterparts to lemmas 1 and 2 they show that depending on the parity of $T^2_1 + T^2_3$ an alternating or a non-alternating choice of jk and yz force an intersection of the arcs γ_{jk} and γ_{yz}. For the proofs note that reflection at the y-axis keeps class $[M]$ invariant but exchanges the first and the second arc, the relevant effect on the pattern is $T^1_1 \leftrightarrow T^2_4$ and $T^1_3 \leftrightarrow T^2_2$.

Lemma 3 *If $T^2_2 = T^2_4$ and $x < j < y < k < z$, then there is an intersection between the arcs γ_{jk} and γ_{yz}.*

Lemma 4 *If $T^2_2 \neq T^2_4$ and $x < j < y < z < k$, then there is an intersection between the arcs γ_{jk} and γ_{yz}.*

The table below shows that it is possible to select ijk and xyz out of six numbers such that the positions of ij and xy respectively jk and yz are any combination of alternating and non-alternating. Hence, according to the lemmas and still assuming that the class is $[M]$ we have at least two intersections between 3-segments γ_{ijk} and γ_{xyz} chosen appropriately depending on the entries of pattern T. We represent elements of ijk by a box □ and elements of xyz by circles •.

□ • □ • □ •	$[T^1_1 = T^1_3$ and $T^2_2 = T^2_4]$
□ • □ • • □	$[T^1_1 = T^1_3$ and $T^2_2 \neq T^2_4]$
• □ □ • □ •	$[T^1_1 \neq T^1_3$ and $T^2_2 = T^2_4]$
• □ □ • • □	$[T^1_1 \neq T^1_3$ and $T^2_2 \neq T^2_4]$

As an example: If the class is $[M]$ and $T = (1, \star, 1, \star, \star, 0, \star, 1)$, where the $\star$'s can be filled arbitrarily, then $T^1_1 = T^1_3$ and $T^2_2 \neq T^2_4$. From the table we see that we have to choose two 3-segments γ_{ijk} and γ_{xyz} such that the order is $i < x < j < y < z < k$ to obtain 3-segments intersecting at least twice.

The cases for the other classes $[L]$ and $[R]$ can be dealt with similarly. This yields a proof of the Conjecture of [2].

References

[1] J. Chalopin, D. Gonçalves, and P. Ochem, *Planar graphs are in 1-STRING*, Proc. of the 18th ACM-SIAM Symp. on Discr. Alg., SODA, (2007), pp. 609–617.

[2] C. Dangelmayr and S. Felsner, *Chordal graphs as intersection graphs of pseudosegments*, in Proceedings of Graph Drawing 2006, vol. 4372 of Lect. Notes Comput. Sci., Springer Verlag, 2006.

[3] N. de Castro, F. J. Cobos, J. C. Dana, A. Márquez, and M. Noy, *Triangle-free planar graphs as segment intersection graphs*, J. Graph Algor. and Appl., 6 (2002), pp. 7–26.

[4] G. Ehrich, S. Even, and R. E. Tarjan, *Intersection graphs of curves in the plane*, J. Combin. Theory Ser. B, 21 (1976), pp. 8–20.

[5] F. Gavril, *The intersection graphs of subtrees in trees are exactly the chordal graphs*, J. Combin. Theory Ser. B, 16 (1974), pp. 47–56.

[6] J. Kratochvíl, *A special planar satisfiability problem and a consequence of its NP-completeness*, Discr. Appl. Math., 52 (1994), pp. 233–252.

[7] E. R. Scheinerman, *Intersection classes and multiple intersection parameters of graphs*, PhD thesis, Princeton University, 1984.

Augmenting the Connectivity of Planar and Geometric Graphs

Ignaz Rutter* Alexander Wolff†

Abstract

In this paper we study some connectivity augmentation problems. Given a connected graph G with some property $\mathcal{P}$, we want to make G 2-vertex connected (or 2-edge connected) by adding edges such that the resulting graph keeps property $\mathcal{P}$. The aim is to add as few edges as possible. The property that we consider is planarity, both in an abstract graph-theoretic and in a geometric setting.

We show that it is NP-hard to find a minimum-cardinality augmentation that makes a planar graph 2-edge connected. For making a planar graph 2-vertex connected this was known. We further show that both problems are hard in the geometric setting, even when restricted to trees. On the other hand we give polynomial-time algorithms for the special case of convex geometric graphs.

We also study the following related problem. Given a plane geometric graph G, two vertices s and t of G, and an integer k, how many edges have to be added to G such that G contains k edge- (or vertex-) disjoint s–t paths? For $k = 2$ we give optimal worst-case bounds; for $k = 3$ we characterize all cases that have a solution.

1 Introduction

Augmenting a given graph to increase its connectivity is important, e.g., for securing communication networks against node and link failures. The planar version of the problem, where the augmentation has to preserve planarity, also has applications in graph drawing [8]. Many graph-drawing algorithms guarantee nice properties (such as convex faces) for graphs with high connectivity. To apply such an algorithm to a less highly connected graph, one adds edges until one reaches the required level of connectivity, uses the algorithm to produce the drawing, and finally removes the added edges again. However, with each removal of an edge one might lose some of the nice properties (such as the convexity of a face). Hence it is natural to look for an augmentation that uses as few edges as possible. Recall that a graph is *k-vertex connected* (*k-edge connected*) if the removal of any subset of $k-1$ vertices (edges) does not make the graph disconnected.

We consider the following two problems.

Planar 2-Vertex Connectivity Augmentation (PVCA):

> Given a connected planar graph $G = (V, E)$ with $n := |V|$ and $m := |E|$, find a smallest set E' of vertex pairs such that the graph $G' = (V, E \cup E')$ is planar and 2-vertex connected (biconnected).

Planar 2-Edge Connectivity Augmentation (PECA) is defined as PVCA, but with *2-vertex connected* replaced by *2-edge connected* (bridge-connected).

The corresponding problems without the planarity constraints have a long history, both for directed and undirected graphs. The unweighted cases can be solved in polynomial time, while the weighted versions are hard [2]. Frederickson and Ja'Ja' [4] gave $O(n^2)$-time factor-2 approximations and showed that augmenting a directed acyclic graph to be strongly connected, and augmenting a tree to be bridge- or biconnected, is NP-complete—even if weights are restricted to the set $\{1, 2\}$. Hsu [5] gave an $O(m + n)$-time algorithm for (unit-weight) 2-vertex connectivity augmentation.

Kant and Bodlaender [8] showed that PVCA is NP-complete and gave 2-approximations for both PVCA and PECA that run in $O(n \log n)$ time. Their 1.5-approximation turned out to be wrong [3]. Fialko and Mutzel gave a 5/3-approximation [3]. Kant showed that PVCA and PECA can be solved in linear time for outerplanar graphs [7].

Provan and Burk [10] considered related problems. Given a planar graph $G = (V, E_G)$ and a planar biconnected (bridge-connected) graph $H = (V, E_H)$ with $E_G \subseteq E_H$, find a smallest set $E' \subseteq E_H$ such that $G' = (V, E_G \cup E')$ is planar and biconnected (bridge-connected). They show that both problems are NP-hard if G is not necessarily connected and give $O(n^4)$-time algorithms for the connected cases.

We also consider a geometric version of the above problems. Recall that a *geometric* graph is a graph where each vertex v corresponds to a point $\mu(v)$ in the plane and where each edge uv corresponds to the straight-line segment $\overline{\mu(u)\mu(v)}$. We are exclusively

*Fakultät für Informatik, Universität Karlsruhe, P.O. Box 6980, D-76128 Karlsruhe, Germany. Supported by grant WO 758/4-3 of the German Science Foundation (DFG). WWW: i11www.ira.uka.de/people/rutter

†Faculteit Wiskunde en Informatica, Technische Universiteit Eindhoven, WWW: www.win.tue.nl/~awolff

problem	planar	outerplanar	geometric	convex
PVCA	NPC [8]	$O(n)$ [7]	NPC	$O(n)$
PECA	NPC	$O(n)$ [7]	NPC	$O(n)$
weighted PVCA	NPC	open	NPC	$O(n^2)$
weighted PECA	NPC	open	NPC	$O(n)$

Table 1: Complexity of PVCA and PECA.

interested in geometric graphs that are *plane*, that is, whose edges intersect at most in their endpoints. Therefore, in this paper by geometric graph we always mean a plane geometric graph. Given a geometric graph G we again want to find a (small) set of vertex pairs such that adding the corresponding edges to G leaves G plane and augments its connectivity.

Rappaport [11] has shown that it is NP-complete to decide whether a set of line segments can be connected to a simple polygon, i.e., geometric PVCA and PECA are NP-complete. Abellanas et al. [1] have shown worst-case bounds for geometric PVCA and PECA. For geometric PVCA they show that $n-2$ edges are sometimes needed and are always sufficient. For geometric PECA they prove that $2n/3$ edges are sometimes needed and $6n/7$ edges are always sufficient. In the special case of plane geometric trees they show that $n/2$ edges are sometimes needed and that $2n/3$ edges are always sufficient for PECA.

Our results. First we show that PECA is NP-complete, too. This answers an open question posed by Kant [6].

Second, we sharpen the result of Rappaport [11] by showing that geometric PVCA and PECA are NP-complete even if restricted to trees.

Third, we give algorithms that solve geometric PVCA and PECA in polynomial time for *convex* geometric graphs, that is, graphs whose vertex sets correspond to point sets in convex position.

Table 1 gives an overview about our results and what has been known previously about the complexity of PVCA and PECA.

Fourth, we consider a related problem, the geometric s–t path augmentation problem. Given a plane geometric graph G, two vertices s and t of G, and an integer $k > 0$, is it possible to augment G such that it contains k edge-disjoint (k vertex-disjoint) s–t paths? We restrict ourselves to $k \in \{2, 3\}$. For $k = 2$ we show that edge-disjoint s–t path augmentation can always be done and needs at most $n/2$ edges. We give an algorithm that computes such an augmentation in linear time. The tree that yields the above-mentioned lower-bound of Abellanas et al. [1] also shows that our bound is tight. For $k = 3$ we show that edge-disjoint s–t path augmentation is always possible, and we give an $O(n^2)$-time algorithm that decides whether a given graph has a vertex-disjoint s–t path augmentation.

2 Complexity results

In this section we show that PECA is NP-complete. This settles an open problem posed by Kant and Bodlaender [8]. Our proof also implies that PVCA is NP-complete, which was already shown by Kant and Bodlaender [8].

Theorem 1 PECA *is NP-complete.*

The hardness proof is by reduction from Planar3SAT, which is known to be NP-hard [9]. The main idea is to use a base graph that is 3-connected (and hence has a unique embedding) and to add some leaves (i.e., degree-1 vertices) to this graph. These leaves can then be embedded in different faces of the graph. It is clear that in order to increase the connectivity the degree of each of the leaves must be enlarged. Ideally (i.e., if the given planar 3SAT formula is satisfiable) the embedding is chosen in such a way that the number of leaves in each face is even, because in this case we need only one edge for every two leaves, which is optimal.

Now we consider geometric PVCA and geometric PECA for connected graphs. These problems are NP-complete as well, however for reasons very different from the planar case. In the geometric setting the embedding is fixed, but two leaves lying in the same face cannot necessarily be connected by a straight-line segment without violating planarity. Especially adding one edge can rule out several others. For example in a square one could add one of the diagonals, but not both. This can again be used to construct a reduction from Planar3SAT.

Theorem 2 *Let G be a plane geometric graph and $k > 0$ be an integer. It is NP-complete to decide whether adding k edges suffices to make G bridge- or biconnected. This is true even if G has exactly $2k$ leaves and G is a tree.*

Once we have shown the result for connected graphs it is easy to extend this to trees. We reduce from the previous case. Let G be a connected plane geometric graph. As long as G contains a cycle, replace an arbitrary edge of a cycle by the construction shown in Figure 1. Call the resulting tree T. Clearly an optimal augmentation connects the two leaves of the construction. Hence an optimal augmentation of T induces an optimal augmentation of G.

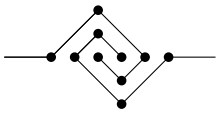


Figure 1: Construction for removing cycles in G.

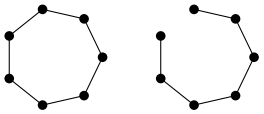


Figure 2: A cycle (left) and a near-cycle (right).

3 Convex geometric graphs

In this section we consider the geometric version of PVCA and PECA in the special case that the input graph is a convex geometric graph. We call an edge *outer edge* if it belongs to the convex hull and *inner edge* otherwise.

Note that PVCA for a convex geometric graph G is trivial: G is biconnected if and only if it contains all edges of the convex hull. Thus we focus on PECA.

If a connected convex geometric graph does not contain an inner edge then it is either a *cycle* or a *near-cycle*, see Figure 2. While the cycle is already bridge-connected, we need a single edge to make the near-cycle bridge-connected.

The basic idea is to decompose an arbitrary convex geometric graph into cycles and near-cycles and use this decomposition to compute an edge set of minimum cardinality that bridge-connects the graph.

Given a convex geometric graph $G = (S, E)$ and an inner edge e of G, we define an operation that we call *splitting G at e*. Splitting G at e yields two subgraphs G_e^+ and G_e^- of G induced by the vertices to the left of or on e and to the right of or on e, respectively.

Lemma 3 *Let G be a connected convex geometric graph and let f be an outer edge of G. If G is not a cycle or a near-cycle, then there exists an inner edge e such that G_e^+ or G_e^- is a cycle or a near-cycle that does not contain f.*

We use Lemma 3 to repeatedly cut off (near-) cycles (i.e., remove all its edges except the split-edge) from G. Each time we cut off a near-cycle, we add to our augmentation the edge that completes the cycle. This is optimal since no edge can cross the split-edge and hence both sides can be processed independently.

If an endpoint of the split-edge is a leaf, we can also remove the split-edge. Otherwise we mark the split-edge and remove it as soon as one of its endpoints becomes a leaf. This ensures that every near-cycle that is cut off actually requires an additional edge in the augmentation. Let's summarize the above.

Theorem 4 *Let $S \subset \mathbb{R}^2$ be a set of n points and let $G = (S, E)$ be a connected convex geometric graph. If the convex hull of S and the corresponding embedding of G are given, we can compute in $O(n)$ time and space a set E' of vertex pairs of minimum cardinality such that $G' = (S, E \cup E')$ is bridge-connected.*

By combining the previous approach with dynamic programming we can also solve the case where each pair of vertices has a positive weight and the aim is to minimize the total weight of the augmentation. Time and space consumption become quadratic.

4 s–t path augmentation

In this section we consider the following problems: Given a plane geometric graph $G = (S, E)$, two vertices $s \neq t$ of G, and an integer $k > 0$, find a smallest set E' of vertex pairs such that $G' = (S, E \cup E')$ is plane and contains k edge-disjoint s–t paths. We also consider the corresponding problem for vertex-disjoint s–t paths. We treat the cases $k = 2$ and $k = 3$.

4.1 Path augmentation for k = 2

The case $k = 2$ is a relaxed version of PECA and PVCA. Although the problem is very restricted in comparison to full 2-edge or 2-vertex connectivity augmentation, it does not seem to be much easier. Like Abellanas et al. [1] we consider the corresponding worst-case problem: how many edges are needed for an s–t path augmentation in the worst case?

Let's quickly discuss the vertex-disjoint case. For the lower bound we can re-use the example of Abellanas et al. [1]: a zig-zag path with end vertices s and t whose vertices are in convex position. There, $n - 2$ edges are needed to establish two vertex-disjoint s–t paths. On the other hand it is not hard to see that $n - 2$ edges always suffice.

Now let's turn to the more interesting edge-disjoint case. Here, the zig-zag path yields a lower bound of $n/2$. Note that the solution actually makes the graph bridge-connected. In fact, Abellanas et al. [1] conjecture that any geometric n-vertex tree can be made bridge-connected by adding at most $n/2$ edges. We show that there is always an s–t path augmentation with at most $n/2$ edges, which is tight by the zig-zag example. We also give a simple algorithm that finds such an augmentation in linear time.

Lemma 5 *Let $S \subseteq \mathbb{R}^2$, let $G = (S, E)$ be a connected plane geometric graph, and let $G' = (S, E')$ with $E \subseteq E'$ be any plane geometric graph that contains G. If s and t are two vertices of G, and G' contains a path of length ℓ between s and t, then there exists an s–t path augmentation of G with at most ℓ edges.*

The proof shows how the path in G' can be used to determine an augmentation for G. The most interesting case is that the path in G' uses an edge that actually is a bridge b in G. The crucial step is to show that we can add a suitable edge to G that induces a cycle with b on it, i.e., b is no longer a bridge. In all other cases we either do not need to add an edge, or we can use the edge of the path.

Let $G = (S, E)$ be a geometric graph. A *triangulation of* G is a triangulation $T = (S, E')$ of the convex hull of S with $E \subseteq E'$. It is well known that every geometric graph can be triangulated [1]. We show:

Lemma 6 *Let $S \subset \mathbb{R}^2$ be a set of n points and let $T = (S, E)$ be a triangulation of the convex hull of S. Then the diameter of T is at most $n/2$.*

The basic idea is to consider growing neighborhoods of the vertices s and t. The *i-th iterated neighborhood* $N_i(v)$ of a vertex v contains all vertices of G within (graph-theoretic) distance at most i from v. Let k be the smallest integer such that $N_k(s) \cap N_k(t) \neq \emptyset$. We use a lower bound on the size of i-th iterated neighborhoods and the fact that $N_{k-1}(s) \cap N_{k-1}(t) = \emptyset$ in order to obtain an upper bound on k. Since G contains an s–t path of length at most $2k$, this upper bound implies the claim.

Lemmas 5 and 6 immediately yield the following.

Theorem 7 *Let $S \subset \mathbb{R}^2$, let $G = (S, E)$ be any plane connected geometric graph with n vertices, and let s and t be any two vertices of G. Then there always exists a set E' of at most $n/2$ pairs of points in S such that $G' = (S, E \cap E')$ is again a plane geometric graph and contains two edge-disjoint s–t paths. Such a set can be computed in linear time.*

This bound can be improved if the convex hull of S does not contain too many points. Take any triangulation and consider the iterated neighborhoods of s and t. As long as a neighborhood has not reached the convex hull it grows by at least three vertices with every iteration. As soon as both neighborhoods have reached the convex hull, we can connect them by a path along the convex hull.

Lemma 8 *The diameter of a plane triangulation $T = (S, E)$ is at most $2(n+3)/5 + h/2$, where $n = |S|$ and h is the number of vertices on the convex hull of S.*

This improves Lemma 6 for $h < (n - 12)/5$.

4.2 Path augmentation for k = 3

We first consider the problem of finding three *vertex*-disjoint s–t paths in a geometric graph. Let $T = (S, E)$ be any plane geometric triangulation and let s and t be any two vertices of T. An edge connecting two vertices of the convex hull that does not belong to the convex hull itself is called a *chord*. A chord uv is (s, t)-*separating* if s and t are in different connected components of $T \setminus uv$.

Obviously T has three vertex-disjoint s–t paths if and only if T does not contain an (s, t)-separating chord. Hence we can rephrase our original question as follows: does any plane geometric graph G without (s, t)-separating chords have a triangulation without (s, t)-separating chords? Such a triangulation would then contain the desired augmentation.

Theorem 9 *Let $S \subset \mathbb{R}^2$ be a set of n points, let $G = (S, E)$ be a connected plane geometric graph, and let $s \neq t$ be two vertices of G. If G contains no (s, t)-separating chord, then there is a triangulation T_G of G that contains three vertex-disjoint s–t paths. Such a triangulation can be computed in $O(n^2)$ time.*

Now we consider the problem of finding three *edge*-disjoint s–t paths. For triangulations we have the following characterization.

Theorem 10 *Let $T = (S, E)$ be a triangulation of the convex hull of S and let $s, t \in S$. Then T contains three edge-disjoint s–t paths if and only if s and t have degree at least 3.*

It is easy to check whether a given geometric graph can be triangulated in such a way.

References

[1] M. Abellanas, A. García, F. Hurtado, J. Tejel, and J. Urrutia. Augmenting the connectivity of geometric graphs. *Comput. Geom. Theory Appl.*, 2008. Appeared online at `http://dx.doi.org/10.1016/j.comgeo.2007.09.001`.

[2] K. P. Eswaran and R. E. Tarjan. Augmentation problems. *SIAM J. Comput.*, 5(4):653–665, 1976.

[3] S. Fialko and P. Mutzel. A new approximation algorithm for the planar augmentation problem. In *Proc. 9th Annu. ACM-SIAM Sympos. Discrete Algorithms (SODA'98)*, pages 260–269, 1998.

[4] G. N. Frederickson and J. Ja'Ja'. Approximation algorithms for several graph augmentation problems. *SIAM J. Comput.*, 10(2):270–283, 1981.

[5] T. Hsu. Simpler and faster biconnectivity augmentation. *J. Algorithms*, 45(1):55–71, 2002.

[6] G. Kant. *Algorithms for Drawing Planar Graphs*. PhD thesis, University of Utrecht, 1993.

[7] G. Kant. Augmenting outerplanar graphs. *J. Algorithms*, 21(1):1–25, 1996.

[8] G. Kant and H. L. Bodlaender. Planar graph augmentation problems. In F. Dehne, J.-R. Sack, and N. Santoro, editors, *Proc. 2nd Workshop Algorithms and Data Structures (WADS'91)*, volume 519 of *Lecture Notes Comput. Sci.*, pages 286–298. Springer-Verlag, 1991.

[9] D. Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.*, 11(2):329–343, 1982.

[10] J. S. Provan and R. C. Burk. Two-connected augmentation problems in planar graphs. *J. Algorithms*, 32:87–107, 1999.

[11] D. Rappaport. Computing simple circuits from a set of line segments is NP-complete. *SIAM J. Comput.*, 18(6):1128–1139, 1989.

Colour Patterns for Polychromatic Four-Colourings of Rectangular Subdivisions

Herman Haverkort* Maarten Löffler† Elena Mumford* Matthew O'Meara‡ Jack Snoeyink§
Bettina Speckmann*

Abstract

A non-degenerate rectangular subdivision is a subdivision of a rectangle into a set of non-overlapping rectangles S, such that no four rectangles meet in a point. We consider a problem that Katz and colleagues call strong polychromatic four-colouring: Colouring the vertices of the subdivision with four colours, such that each rectangle of S has all colours among its four corners. By considering the possible colouring patterns, we can give short constructive proofs of colourability for subdivisions that are *sliceable* or *one-sided*. We also present techniques and observations for non-sliceable, two-sided subdivisions.

1 Introduction

A *rectangular subdivision* is a set S of rectangles with disjoint interiors whose union is a rectangle $r(S)$. The set of *vertices* of S is the union of the sets of vertices (corners) of the rectangles in S. If S contains four rectangles that meet in a single vertex, we say that S is *degenerate*. A *non-degenerate rectangular subdivision* is a rectangular subdivision in which each vertex is a corner of only one or two rectangles. Unless otherwise specified, a *subdivision* is a non-degenerate rectangular subdivision.

Dinitz et al. [1] showed that it is possible to colour the vertices of any subdivision S with three colours so that each rectangle in S is *polychromatic*—has at least one vertex of each colour. They conjectured that this is also possible with four colours. This conjecture is in fact a special case of a much older conjecture by Seymour [6] concerning the edge-colouring of a special class of planar graphs, so-called $4-graphs$. Seymour's conjecture was proven by Guenin [3]. We are thankful to Dimitrov et al. [2] for pointing out that Guenin's result implies that each non-degenerate subdivision has indeed a strong polychromatic four-colouring.

*Department of Mathematics and Computer Science, Eindhoven University of Technology, cs.herman@haverkort.net, e.mumford@tue.nl, speckman@win.tue.nl

†Department of Information and Computing Sciences, Utrecht University, loffler@cs.uu.nl

‡Department of Mathematics, University of Chicago, mattjomeara@gmail.com

§Department of Computer Science, University of North Carolina at Chapel Hill, snoeyink@cs.unc.edu

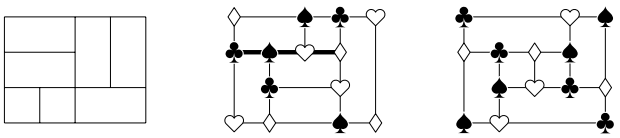


Figure 1: From left to right: a degenerate subdivision that is not colourable; a sliceable subdivision (but not one-sided, because of the fat segment); a one-sided (but not sliceable) subdivision.

Guenin's proof is non-constructive. In the remainder of this abstract we focus on constructive proofs for specialized subdivisions and report on observations concerning so-called colour patterns. For any subdivision S let a *colouring pattern* of S be an assignment of four colours to the vertices of S such that each rectangle is polychromatic. We say S is *colourable* if it admits at least one colour pattern. Let a *boundary colouring pattern* of S be the restriction of a colouring pattern of S to the corners of the boundary of S.

Known and new results. Not all degenerate rectangular subdivisions are colourable (see Figure 1).

Sliceable subdivisions. A subdivision is called *sliceable* if it can be obtained by recursively slicing a rectangle with horizontal and vertical lines. Horev et al. [5] call these *guillotine* subdivisions, and show that they are always colourable. We give a short proof using *boundary colour patterns*.

Canonical form. We can order the rectangles of a subdivision so that prefixes form monotone staircases; for some subdivisions this order is unique. We show that any colouring pattern of a subdivision can be realized by a subdivision with a unique order, and give a procedure to convert any given subdivision into such a *canonical form*.

One-sided subdivisions. A *maximal line segment of* S is a line segment that is completely covered by edges of the rectangles of S for which no extension is covered. A subdivision is *one-sided* if and only if, for every maximal line segment s, the vertices in the interior of s are T-junctions that all have the leg on the same side of s. We will prove that every one-sided subdivision is colourable.

Other subdivisions. Here we present some observations regarding possible algorithms or hypothetical counterexamples. We call a subdivision *atomic* or *semi-atomic* if every proper subset $S' \subset S$ such

that S' is a subdivision, consists of only one rectangle, or at most two rectangles, respectively. We show that if there is a subdivision that is not colourable, the smallest such subdivision must be semi-atomic, or semi-atomic with one additional rectangle glued to an external edge. We also study sets of boundary colour patterns of different subdivisions of staircases which are surprisingly dissimilar.

2 Corner colour patterns

Consider any subdivision S, whose union is a rectangle $r(S)$. We say S is even if $|S|$ is even, and S is odd if $|S|$ odd.

Lemma 1 *If S is odd and colourable, then every boundary colouring pattern is polychromatic.*

If S is even and colourable, then in every boundary colouring pattern of S, either all four corners have the same colour or two corners have the same colour and two corners have another colour.

Proof. Number the colours to be used from 1 to 4. For a given colouring pattern of S, let C_c be the number of corners of $r(S)$ with colour c, and let I_c the number of other vertices with colour c. Note that each of the corners of $r(S)$ gives its colour to exactly one rectangle of S, while the remaining vertices of S give their colour to exactly two rectangles of S. No rectangle can have two corners of the same colour. Thus, for every colour c, we have $C_c + 2I_c = |S|$.

When S is odd, C_c must be odd, and therefore positive for each colour. Since $\sum_{c\in\{1,2,3,4\}} C_c = 4$, this implies that $C_c = 1$ for each $c \in \{1, 2, 3, 4\}$.

When S is even, C_c must be even, and therefore either 0, 2 or 4, for each colour. □

Subdivisions thus allow five boundary colouring patterns:

- ⁚⁚ all corners have different colours;
- ⁼ corners use two colours, paired horizontally;
- ‖ corners use two colours, paired vertically;
- × corners use two colours, paired diagonally;
- □ all corners have the same colour.

3 Sliceable subdivisions

Theorem 2 *Sliceable subdivisions are colourable.*

Proof. We prove by induction on the number of rectangles, that every odd sliceable subdivision with $|S| \leq n$ can be coloured and must have ⁚⁚ as its only boundary colouring pattern, and every even sliceable subdivision with $|S| \leq n$ can be coloured and has at least two of {⁼, ‖, ×} as boundary colouring patterns.

When $n = 1$, S has boundary colouring pattern ⁚⁚.

Now, consider a subdivision with $|S| = n + 1$ composed of two subdivisions, L and R, separated by a vertical line (the case of a horizontal separating line is symmetric). By induction, both L and R can be coloured separately. We distinguish four cases, depending on whether L and R are odd or even.

(i) If L and R are both odd, then, if necessary, we relabel the colours of R to match the two corners shared with L. The corners of $r(S)$ now use the remaining two colours. We may swap these colours in R so on the boundary they are paired either horizontally ⁼ or diagonally ×, satisfying the induction hypothesis.

(ii) If L is even and R is odd, then S is odd. By induction, L has as a boundary colouring pattern ⁼ or ×. We can recolour R to match L on shared vertices, resulting in a ⁚⁚ pattern for the boundary of S.

(iii) The case of L odd and R even is symmetric.

(iv) For the final case, in which L, R and S are all even, we use the following notation for composing boundary colouring patterns: $P_L P_R \to P_S$ means that joining boundary colouring pattern P_L for L with P_R for R gives boundary colouring pattern P_S for S.

If L and R are both even and admit the ‖ pattern, then S admits the pattern ‖, (‖‖ → ‖). Furthermore, by induction, both L and R admit at least one more pattern out of ⁼ and ×. Since ⁼⁼ → ⁼, ×× → ⁼, ⁼× → ×, and ×⁼ → ×, we have that S has as a pattern at least one of ⁼ and ×.

If L and R are both even and L does not admit the ‖ pattern, then, by induction, L admits *both* the ⁼ and × patterns. R admits at least one of ⁼ and ×. Thus we can obtain the boundary colouring patterns ⁼⁼ → ⁼ and ×⁼ → ×, or ⁼× → × and ×× → ⁼; in both cases we obtain patterns ⁼ and × for S.

If L and R are both even and R does not admit the ‖ pattern, we apply the above arguments symmetrically and again S admits ⁼ and × as boundary colouring patterns.

Therefore the theorem holds for S by induction. □

4 Canonical form

One can build up a subdivision by listing rectangles $R_1, \ldots, R_n$ such that, for any i, the rectangles $R_1, \ldots, R_i$ cover the rectangle defined by the lower right corner of S and the upper left corner of R_i. That is, the rectangles with indices $\leq i$ are separated from those with index $> i$ by a *staircase* that is monotonically increasing in x and y. Such an ordering can be found for any subdivision by rotating it clockwise and extending the "aboveness" partial order to a total order [4]. We say that a subdivision has a *canonical ordering* if there is a unique extension. We can use aboveness to convert a subdivision into a *canonical form* – a subdivision with a unique extension.

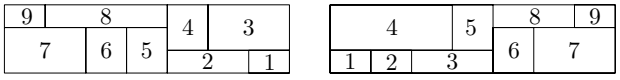


Figure 2: A subdivision in ↖-order and ↗-order.

Lemma 3 *One can in $O(n \log n)$ time convert a subdivision into canonical form – having unique ordering – without changing the colouring patterns.*

Proof. [Sketch] Consider the rotated subdivision S as a collection of open rectangles, and open *maximal line segments* (i.e. not containing their endpoints, but extending vertically or horizontally as far as possible – ending at T-junctions.) These are convex and disjoint; any set of disjoint convex objects can be given a total order consistent with aboveness for the direction from upper left to lower right corners of S.

Now, for the vertical and horizontal maximal segments, replace the x and y coordinates, respectively, by their ranks, redraw the subdivision and rotate it back, as in Figure 2. Including the segments in the ordering makes the ordering unique, and the transformation builds that uniqueness into the subdivision. This transformation does not affect the bipartite graph in which each rectangle connects to the vertices at its four corners. □

5 One-sided subdivisions

Theorem 4 *One-sided subdivisions are colourable.*

Proof. We may assume that $S = \{R_1, \ldots, R_n\}$ is a one-sided subdivision in canonical form (↖-order), since conversion to canonical form preserves one-sidedness. We claim that any two consecutive rectangles in ↖-order share a corner: Assume inductively this claim holds after adding R_{i-1}. Up to reflection we may assume we are adding R_i above R_{i-1}; let h be the maximal line segment that contains the top edge of R_{i-1}. Since S is in ↖-order, the left endpoint $l(h)$ of h must be the top left corner $tl(R_{i-1})$ of R_{i-1}, and the right endpoint $r(h)$ of h must be the bottom right corner of R_i. If $r(h)$ is also the top right corner $tr(R_{i-1})$ of R_{i-1}, then R_i and R_{i-1} share that corner. Otherwise $tr(R_{i-1})$ is a downward T-junction on h. Since S is one-sided, the bottom left corner $bl(R_i)$ of R_i cannot be an upward T-junction on h, so we have $bl(R_i) = l(h) = tl(R_{i-1})$, proving our claim.

We now define a path through S that we can use to colour S. Consider the n rectangles of S in ↖-order, $R_1, \ldots, R_n$. They define a path $u_0, \ldots, u_n$ as follows: let u_0 be the lower right corner of R_1, let u_n be the upper left corner of R_n, and let u_i (for $0 < i < n$) be the corner shared by R_i and R_{i+1} (in case of a tie, the left- and bottommost corner common to R_i and R_{i+1} is chosen). Symmetrically, we define a path $v_0, \ldots, v_n$ from the lower left corner to the upper right corner of S, following the rectangles in ↗-order – the canonical order of their horizontal mirror image. Ties are now broken in favour of the right- and topmost corner shared by two rectangles.

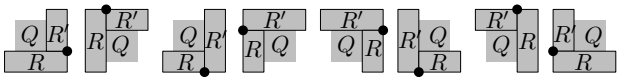


Figure 3: In all cases, Q appears between R and R' either in ↖-order or in ↗-order.

We claim that these two paths are vertex-disjoint. Suppose, for the sake of contradiction, that there is a vertex w that appears on both paths. Then the two rectangles R and R' of which w is a corner must be adjacent in both ↖- and ↗-order. Moreover, R and R' share only one corner, otherwise the tie-breaking mechanism would have put w in one path and the other shared corner in the other path. Now consider all ways in which R and R' can share exactly one corner. One can verify (see Figure 3) that in at least one of the two orderings, R and R' are *not* adjacent. This contradicts our assumption, proving our claim.

We now colour u_i black and v_i red for even i, and we colour u_i white and v_i green for odd i. Since each rectangle has two successive corners on each path, this ensures that each rectangle is polychromatic. □

6 On the hypothetical smallest counterexample

Lemma 5 *If S is even and colourable, it allows at least one boundary colouring pattern out of ⊏ and ⫿, and at least one pattern out of × and □.*

Proof. Consider a subdivision with boundary colour pattern □ and assume that all corners are coloured black. Consider the graph whose nodes are the vertices of S that are coloured black or white, and whose arcs are given by the pair of the black corner and the white corner of each rectangle in S. This graph consists of two paths whose four end nodes are the corners of S, and possibly a number of cycles; if on any of these paths or cycles we swap all black and white vertices, we maintain a valid colouring pattern.

Since each rectangle contains only one arc, the two paths cannot cross inside a rectangle; since each vertex has degree at most two, the two paths cannot cross in a vertex either. So the path that starts in the lower left corner of S ends in either the upper left or the lower right corner of S (never in the diagonally opposite corner). In the first case, we can change the □-pattern into a ⫿-pattern by swapping the colours on that path; in the second case, we can change the □-colouring into a ⊏-colouring by swapping the colours on that path—see Figure 4. Note that by swapping colours in this way, every rectangle is either left unchanged (if the arc defined by it is not on the swapping

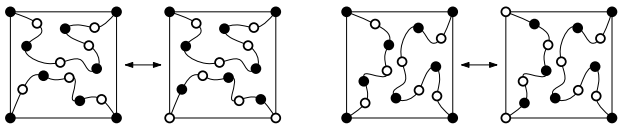


Figure 4: Changing colour patterns into others.

path) or it has its black corner turned white and its white corner turned black, so that each rectangle remains polychromatic. Hence every subdivision that admits a □-pattern also admits a ⊐ or a ▯▯ pattern.

With similar arguments we can show that a × pattern can always be changed into a ⊐ or ▯▯, and that ⊐ and ▯▯ can always be changed into □ or ×. □

For any subdivision S let the set of boundary colouring patterns that S admits be $P(S)$. Then $P(S)$ is one of the eleven sets: $\emptyset$, {⊐, □}, {▯▯, □}, {▯▯, ×}, {⊐, ×}, {▯▯, ×, □}, {⊐, ×, □}, {▯▯, ⊐, ×}, {▯▯, ⊐, □}, {▯▯, ⊐, ×, □}, and {∷}. Figure 5 shows the smallest subdivisions for the last eight sets.

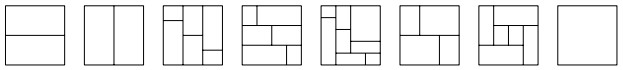


Figure 5: Subdivisions for each good pattern set.

Observe that if there exists a subdivision S such that $P(S) = \{⊐, □\}$, for every colouring pattern of S, the top corners of S will have the same colour. We can extend S to a subdivision S' by gluing an a rectangle R across the top of S. Then S' is not colourable as every colouring pattern of S forces R to be non-polychromatic. Hence, if any subdivision has $P(S) = \{⊐, □\}$ or (by symmetry) $P(S) = \{▯▯, □\}$, then there are subdivisions that are not colourable. Therefore, we say that $\emptyset$, {▯▯, □} and {⊐, □} are *bad* pattern sets and the other pattern sets are *good*.

Theorem 6 *Suppose there exist subdivisions with bad boundary colouring pattern sets. Let S be a smallest such subdivision. Then S does not contain any proper subset T whose union forms a rectangle with $|T| > 2$, that is, S is semi-atomic.*

Proof. [Sketch] We claim that if S contains a proper rectangular subset T with $|T| > 2$, then we can construct a subdivision S' with $|S'| < |S|$, such that each colouring pattern of S' can be transformed into a colouring pattern of S with the same boundary colouring pattern. Since S is the smallest subdivision with a bad pattern set, the smaller subdivision S' must have a good pattern set. Now, since $P(S') \subseteq P(S)$, and no superset of a good pattern set is bad, S must have a good boundary colouring pattern set. But, this contradicts our assumption that S is a smallest subdivision with a bad pattern set.

To prove our claim we show how to construct S' and transform a colouring pattern for S' into a colouring pattern for S. If $P(T) = \{∷\}$, then let S' be the subdivision obtained from S by replacing T with a single rectangle T'. Consider a colouring pattern of S'. By definition of a valid colouring, T' must be polychromatic. Since ∷ ∈ $P(T)$, we can remove T' from S' again, colour T with the same colours on its corners as T', and insert T in S'. Thus we obtain a colouring pattern for S, such that $P(S) = P(S')$. If $P(T)$ differs from {∷}, similar constructions are possible: if $P(T) \supseteq \{⊐, ×\}$, we replace T by two rectangles separated by a vertical line; if $P(T) \supseteq \{▯▯, ×\}$, we replace T by two rectangles separated by a horizontal line; if $P(T) \supseteq \{⊐, ▯▯, □\}$, we shrink T to a line segment. □

7 Looking for algorithms and counterexamples

Trying to generalize the argument for colouring sliceable subdivisions, we consider cutting a subdivision S in ↖-order into a prefix set of rectangles S_1 and a postfix set of rectangles S_2 and colouring each separately. Let $P(S_1)$ be the set of boundary colouring patterns along the cut where each pattern is relabeled to be in lex min order. Let S_2' be S_2, reflected in the line $x = y$. We say that $P(S_1)$ and $P(S_2')$ *couple* with pattern $p \in P(S_1)$ if S_2 can be relabeled so that S is colourable with pattern p along the cut.

We enumerated sets of rectangles that may arise as a prefix in a subdivision in ↖-order. For stairs with four corners, 359 distinct stair colouring pattern sets were found. Of the possible 15 patterns the min set size was 3 while the max was 14 with average 8.6. For each pair of stair colouring pattern sets, the min size of the coupling set was 1 while the max was 14 with average 5.5. For each stair pattern we found a pair where that was the only pattern they coupled along. In the light of these results Guenin's intricate proof is particularly impressive.

References

[1] Y. Dinitz, M. J. Katz, and R. Krakovski. Guarding rectangular partitions. In *Abstr. 23rd Europ. Workshop Comp. Geom.*, pages 30-33, 2007.

[2] D. Dimitrov, E. Horev, and R. Krakovski. Polychromatic 4-colorings of rectangular partitions. In *Abstr. 24th Europ. Workshop Comp. Geom.*, 2007 (to appear).

[3] B. Guenin. Packing T-joins and edge colouring in planar graphs. Manuscript.

[4] L. J. Guibas and F. F. Yao. On translating a set of rectangles. *Computational Geometry* 1:61–77, 1983.

[5] E. Horev, M. J. Katz, M. Löffler, and R. Krakovski. Polychromatic 4-coloring of guillotine subdivisions. Manuscript.

[6] P. D. Seymour. On multi-colourings of cubic graphs, and conjectures of Fulkerson and Tutte. *Proc. London Math. Soc.* s3-38(3):423-460, 1979.

Polyhromatic 4-Colorings of Rectangular Partitions

Darko Dimitrov* Elad Horev† Roi Krakovski†

Abstract

A *rectangular partition* is a partition of a plane rectangle into an arbitrary number of non-overlapping rectangles such that no four rectangles share a corner. In this note, it is proven that every rectangular partition admits a vertex coloring with four colors such that every rectangle, except possibly for the outer rectangle, has all four colors on its boundary. This settles a conjecture of Dinitz et al. [3]. The proof is short, simple and based on 4-edge-colorability of a specific class of planar graphs.

1 Introduction

A *polychromatic k-coloring* of a plane graph G is an assignment of k colors to the vertices of G such that each face of G, except for possibly the outer face, has all k colors on its boundary. More formally, a polychromatic k-coloring of a plane graph G is a mapping $\varphi : V(G) \rightarrow \{1, \ldots, k\}$, such that for every internal face of G there exist k vertices $\{u_1, \ldots, u_k\}$ on its boundary such that $\varphi(u_i) = i$, for $i = 1, \ldots, k$. Note that a polychromatic k-coloring allows monochromatic edges. The *polychromatic number* of a plane graph G, namely $\chi_f(G)$, is the *maximum* number k such that G admits a polychromatic k-coloring.

A general and elegant result concerning polychromatic colorings of plane graphs was recently obtained by Alon *et al.* [1]. Let g be the length of a shortest face of a plane graph G. Clearly, $\chi_f(G) \leq g$. Alon *et al.* proved that for any plane graph G, $\chi_f(G) \geq \lfloor (3g-5)/4 \rfloor$, and showed that this bound is sufficiently tight by presenting plane graphs G for which $\chi_f(G) \leq \lfloor (3g+1)/4 \rfloor$. In addition, they proved that for a plane graph G, determining whether $\chi_f(G) \geq 3$ is NP-hard.

Mohar and Škrekovski [9] proved that every simple plane graph admits a polychromatic 2-coloring. Their proof is short and relies on the four-color theorem. Bose *et al.* [2] provided an alternative proof that does not rely on the four-color theorem. Hoffmann and Kriegel [5] proved that every 2-connected bipartite plane graph can be transformed into an Eulerian triangulation by adding edges only. Since every plane Eulerian triangulation is 3-colorable in the regular sense [11], it follows that every 2-connected bipartite plane graph admits a polychromatic 3-coloring. Horev and Krakovski [8] proved that every plane graph of degree at most 3, other than K_4 (the complete graph on four vertices), admits a polychromatic 3-coloring. Finally, Horev *et al.* [6] proved that every 2-connected cubic bipartite plane graph admits a polychromatic 4-coloring. This result is tight, since any such graph must contain a face of size four.

A *rectangular partition* is a partition of a plane rectangle into an arbitrary number of non-overlapping rectangles, such that no four rectangles meet at a common vertex (see Fig. 1 for an illustration). The *order* of a rectangular partition is the number of rectangles in the partition including the outer rectangle. One may view a rectangular partition as a plane graph whose vertices are the corners of the rectangles and edges are the line segments connecting these corners. Consequently, we refer to the corners of a rectangular partition as its vertices.

A rectangle r of a rectangular partition R may have numerous vertices of R on its boundary. However, the rectangle r is defined by a set of exactly four vertices of R, denoted $D(r)$. Two rectangles r_1 and r_2 of R are said to be *incident* in R if $D(r_1) \cap D(r_2) \neq \emptyset$. Moreover, every vertex u of R such that $u \in D(r_1) \cap D(r_2)$ is called a *common incidence vertex* of r_1 and r_2. Note that every vertex of R is a common incidence vertex between some two rectangles of R.

A stronger extension of polychromatic 4-colorings of rectangular partitions is to require a coloring that for every rectangle all four colors appear on the four vertices defining it. More formally, we define a *strong polychromatic 4-coloring* of a rectangular partition R as a vertex coloring of R with four colors such that every rectangle r of R has all four colors appearing in the vertex set $D(r)$.

Guillotine subdivisions are a well-studied subfamily of rectangular partitions. Horev *et al.* [7] showed that every guillotine subdivision admits a strong polychromatic 4-coloring.

Dinitz *et al.* [3] proved that every rectangular partition admits a polychromatic 3-coloring, and conjectured that every rectangular partition admits a polychromatic 4-coloring. In this note, we prove the conjecture raised by Dinitz *et al.* in [3]. Actually, we prove a stronger claim by showing that every rect-

*Institut für Informatik, Freie Universität Berlin, Takustrasse 9, D-14195 Berlin, Germany, darko@inf.fu-berlin.de

†Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel, {horevel, roikr}@cs.bgu.ac.il

angular partition admits a strong polychromatic 4-coloring. Our proof is short and simple, and is based on 4-edge-colorability of 4-graphs (see Definition 1). The latter result relies on the four-color theorem of planar graphs.

2 r-graphs and polychromatic 4-colorings of rectangular partitions

For a (multi)-graph G we write $V(G)$ to denote the vertex set of G. For a vertex set $X \subseteq V(G)$ the set of edges with one endpoint in X and the other in $V(G) \setminus X$ is called an *edge-cut* of G induced by X, and is denoted $(X, V(G) \setminus X)$.

A *k-edge-coloring* of a (multi)-graph G is an assignment of k colors to the edges of G such that edges that share a common endpoint are assigned distinct colors.

Definition 1 *An r-graph is an r-regular (multi)-graph G on an even number of vertices with the property that every edge-cut which separates $V(G)$ into two sets of odd cardinality has size at least r.*

r-graphs were introduced in 1979 by Seymour [10], who conjectured that every planar 4-graph is 4-edge-colorable. This conjecture was later proved by Guenin [4], who also showed that the corresponding result also holds for 5-graphs.

Our proof that every rectangular partition admits a strong polychromatic 4-coloring relies on the following theorem.

Theorem 1 (Guenin) *Every planar 4-graph is 4-edge-colorable.*

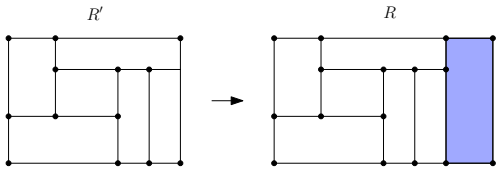


Figure 1: Extending a rectangular partition of odd order to a rectangular partition of even order.

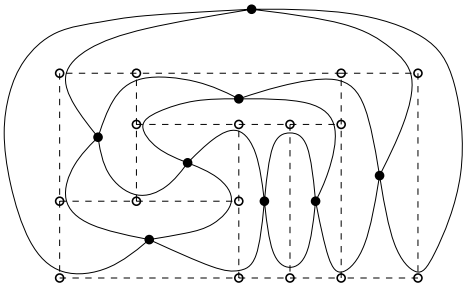


Figure 2: Rectangular partition and its "dual" graph.

Our main result is as follows.

Theorem 2 *Every rectangular partition admits a strong polychromatic 4-coloring.*

Proof. Let R be a rectangular partition. One may assume that R is of even order; for otherwise, add one rectangle to R, obtaining a new rectangular partition of even order. Fig. 1 illustrates this addition. Define G to be the graph obtained from R as follows. To each rectangle of R (including the outer rectangle) assign a vertex in G. For every vertex u of R add an edge between the vertices of G that correspond to the two rectangles of R for which u is a common incidence vertex. Note that G is a 4-regular planar (multi)-graph with an even number of vertices. See Fig. 2 for an illustration of the graph G.

We proceed by showing that G is a 4-graph. We will show that every edge-cut of G consists of at least four edges. For a vertex set $X \subset V(G)$, let $(X, V(G) \setminus X)$ be an edge-cut of G. Consider a maximal connected component, namely C, of $G[X]$ (the subgraph of G induced by X). Observe that in R, the component C corresponds to a union of rectangles whose boundary defines a rectilinear polygon P. Consequently, P contains at least four vertices of R on its boundary which are convex. Consider an edge e in G that corresponds to a convex vertex of P. The edge e does not connect two vertices that correspond to rectangles of P, i.e., the edge e does not connect two vertices in C. Moreover, by the maximality of C, the edge e does not connect two vertices in X. Consequently, e crosses the edge-cut $(X, V(G) \setminus X)$. By Theorem 1, the graph G is 4-edge-colorable. Given a 4-edge-coloring of G, we color every vertex of R with the color of its corresponding edge in G. By the definition of G, the claim follows. □

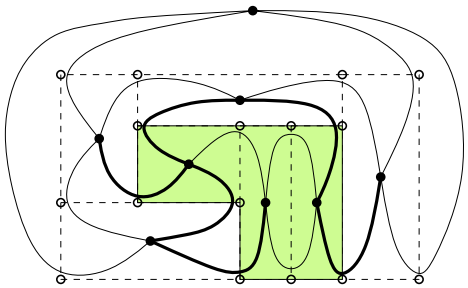


Figure 3: Splitting a rectangular partition in to two subsets of rectangles, and the corresponding edge-cut (fat edges).

Consider a rectangular partition in which it is allowed for four rectangles to share a common corner. It is interesting to note that there are rectangular partitions of this type that do not admit strong a polychromatic 4-coloring. An example of such a partition is shown in Fig. 4. Note, however, that this partition

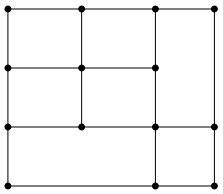


Figure 4: A counterexample.

admits a polychormatic 4-coloring. Hence, we conclude this note with the following question.
Is it true that every rectangular partition, of the latter form, admits a polychromatic 4-coloring?

References

[1] N. Alon, R. Berke, K. Buchin, M. Buchin, P. Csorba, S. Shannigrahi, B. Speckmann, and P. Zumstein. Polychromatic colorings of plane graphs. submitted, 2007.

[2] P. Bose, D. Kirkpatrick, and Z. Li. Worst-case-optimal algorithms for guarding planar graphs and polyhedral surfaces. *Comput. Geom. Theory Appl.*, 26(3):209–219, 2003.

[3] Y. Dinitz, M. J. Katz, and R. Krakovski. Guarding rectangular partitions bounding box of a point set in three dimensions. In *23rd European Workshop on Computational Geometry*, 2007.

[4] B. Guenin. Packing T-joins and edge coloring in planar graphs. submitted.

[5] F. Hoffmann and K. Kriegel. A graph coloring result and its consequences for polygon guarding problems. *SIAM J. Discrete Math.*, 9(2):210–224, 1996.

[6] E. Horev, M. J. Katz, and R. Krakovski. Polychromatic colorings of cubic bipartite plane graphs. submitted, 2007.

[7] E. Horev, M. J. Katz, R. Krakovski, and M. Löffler. Polychromatic 4-colorings of guillotine subdivision. submitted, 2007.

[8] E. Horev and R. Krakovski. Polychromatic colorings of bounded degree plane graphs. submitted, 2007.

[9] B. Mohar and R. Škrekovski. The Grötzsch theorem for the hypergraph of maximal cliques. *Electr. J. Comb.*, R26:1–13, 1999.

[10] P. D. Seymour. On multi-colourings of cubic graphs, and conjectures of Fulkerson and Tutte. *London Math Soc.*, 38:423–460, 1979.

[11] D. West. *Introduction to Graph Theory*. Prentice Hall, Englewood Cliffs, NJ, 1996.

Exact Implementation of Arrangements of Geodesic Arcs on the Sphere with Applications*

Efi Fogel† Ophir Setter† Dan Halperin†

Abstract

Recently, the `Arrangement_2` package of CGAL, the Computational Geometry Algorithms Library, has been greatly extended to support arrangements of curves embedded on two-dimensional parametric surfaces. The general framework for sweeping a set of curves embedded on a two-dimensional parametric surface was introduced in [3]. In this paper we concentrate on the specific algorithms and implementation details involved in the exact construction and maintenance of arrangements induced by arcs of great circles embedded on the sphere, also known as geodesic arcs, and on the exact computation of Voronoi diagrams on the sphere, the bisectors of which are geodesic arcs. This class of Voronoi diagrams includes the subclass of Voronoi diagrams of points and its generalization, power diagrams, also known as Laguerre Voronoi diagrams. The resulting diagrams are represented as arrangements, and can be passed as input to consecutive operations supported by the `Arrangement_2` package and its derivatives. The implementation is complete in the sense that it handles degenerate input, and it produces exact results. An example that uses real world data is included. Additional material is available at `http://www.cs.tau.ac.il/~efif/VOS`.

1 Introduction

Given a finite collection $\mathcal{C}$ of geometric objects (such as lines, planes, or spheres) the *arrangement* $\mathcal{A}(\mathcal{C})$ is the subdivision of the space where these objects reside into cells as induced by the objects in $\mathcal{C}$. In this paper we concentrate on the particular class of arrangements, where the embedding space is the sphere, and the inducing objects are geodesic arcs. There is an analogy between this class of arrangements and the class of planar arrangements induced by linear curves (i.e., segments, rays, and lines), as properties of linear curves in the plane can be often, (but not always), adapted to geodesic arcs on the sphere. The ability to robustly construct arrangements of geodesic arcs on the sphere, and carry out exact operations on them using only (exact) rational arithmetic is a key property that enables an efficient implementation.

Recently, a software package that computes exact arrangements of general circles on the sphere was introduced [5]. The extended `Arrangement_2` package was used to compute arrangements on quadrics [3] and on Dupin cyclides [4], which contain the torus as a special case. The technique to compute Voronoi diagrams on two-dimensional parametric surfaces described in this paper can be applied to these surfaces as well, conditioned on the ability to handle bisectors of sites embedded on these surfaces.

Voronoi diagrams were thoroughly investigated and were used to solve many geometric problems [1, 17]. One of the interesting properties observed about this decomposition of a space is its strong connection to arrangements [6], a property that yields a very general approach for computing Voronoi diagrams.

The concept of computing cells of points that are closer to a certain object than to any other object, among finite number of objects, was extended to various kinds of geometric sites, ambient spaces, and distance functions, e.g., power diagrams of circles in the plane, multiplicatively weighted Voronoi diagrams, additively weighted Voronoi diagrams [1, 2, 17]. One immediate extension is computing Voronoi diagrams on two-dimensional parametric surfaces [12] in general, and on the sphere [15, 16] in particular.

2 Arrangements on surfaces

A parameterized surface S is defined by a function $f_S : \mathbb{P} \to \mathbb{R}^3$, where the domain $\mathbb{P} = U \times V$ is a rectangular two-dimensional parameter space with bottom, top, left, and right boundaries, and the range f_S is a continuous function. We allow $U = [u_{\min}, u_{\max}]$, $U = [u_{\min}, +\infty)$, $U = (-\infty, u_{\max}]$, or $U = (-\infty, +\infty)$, and similarly for V. A *contraction point* $p \in S$ is a singular point, which is the mapping of a whole boundary of the domain $\mathbb{P}$. For example, if the top boundary is contracted, we have $\forall u \in U, f_S(u, v_{\max}) = p'$ for some fixed point $p' \in \mathbb{R}^3$. An *identification curve* $C \subset S$ is a continuous curve, which is the mapping of opposite closed boundaries of the domain $\mathbb{P}$. For example, if the left and right boundaries are identified, we have $\forall v \in V, f_S(u_{\min}, v) = f_S(u_{\max}, v)$. A *curve* in the domain is defined as a function $\gamma : I \to \mathbb{P}$ where (i) I is

*This work has been supported in part by the IST Programme of the EU as Shared-cost RTD (FET Open) Project under Contract No IST-006413 (ACS - Algorithms for Complex Shapes), by the Israel Science Foundation (grant no. 236/06), and by the Hermann Minkowski–Minerva Center for Geometry at Tel Aviv University.

†School of Computer Science, Tel-Aviv University, 69978, Israel. {efif,ophirset,danha}@post.tau.ac.il

an open, half-open, or closed interval with endpoints 0 and 1; (ii) γ is continuous and injective, except for closed curves, where $\gamma(0) = \gamma(1)$; (iii) if $0 \notin I$, the curve has no start point, and emanates "from infinity". It holds that $\lim_{t\to 0+} \|\gamma(t)\| = \infty$ (we have a similar condition if $1 \notin I$), and we assume that these limits exist. A *weakly u-monotone curve* $C \subset S$ is the mapping of a curve γ, such that if $t_1 < t_2$ then $\gamma(t_1)$ is lexicographically smaller than $\gamma(t_2)$.

The `Arrangement_2` package of CGAL, the Computational Geometry Algorithms Library,[1] included in Version 3.3 supports planar arrangements induces by planar curves. Recently, this package has been extended to support arrangements of curves embedded on a two-dimensional parametric surface [3]. The extended package can handle curves that approach a boundary in case it is unbounded, or reach a boundary in case it is bounded. In the bounded case, a boundary can define either a contraction point or an identification curve[2]. The extended package is realized as a prototypical CGAL package, and is planned to be included in the next public release.

The main class of the `Arrangement_2` package represents the embedding of a set of continuous weakly u-monotone curves that are pairwise disjoint in their interiors on a two-dimensional parametric surface. The package offers various operations on arrangements stored in this representation, such as point location, insertion of curves, removal of curves, and overlay computation.

Code reuse is maximized by generalizing the prevalent algorithms and their implementations. The generalized code handles features embedded on a modified surface $\widetilde{S} : f_{\widetilde{S}} = f_S(u,v) \,|\, (u,v) \in \widetilde{\mathbb{P}}$ defined over a modified parameter space $\widetilde{\mathbb{P}}$, where the boundaries are removed. Specific code that handles features that approach or reach the boundaries is added to yield a complete implementation.

The implementation of the various algorithms that construct and manipulate arrangements is generic, as it is independent on the type of curves they handle. All steps of the algorithms are enabled by a minimal set of geometric primitives, such as comparing two points in uv-lexicographic order, computing intersection points, etc. These primitives are gathered in a traits class, which models a *geometry-traits* concept [19]. Different geometry-traits classes are provided in the `Arrangement_2` package to handle various families of curves, e.g., line segments, conic arcs, etc.

The geometry-traits concept is factored into a hierarchy of refined concepts. The refinement hierarchy is defined according to the identified minimal requirements imposed by different algorithms that operate on arrangements, thus alleviating the pro-

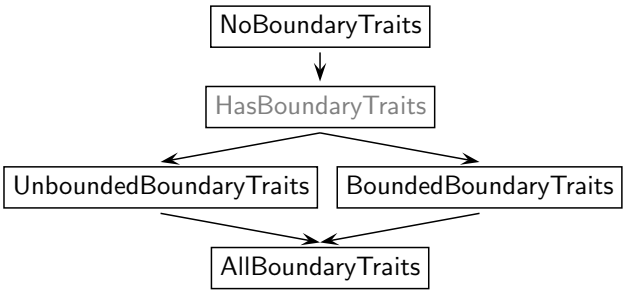


Figure 1: Hierarchy of Geometry Traits Concepts for Arrangement on Surface.

duction of traits classes, and increasing the usability of the algorithms. We refer to the entire hierarchy of refinements defined in Version 3.3 as a single concept called *NoBoundaryTraits* for simplicity. The extended package introduces new concepts, models of which are able to handle unbounded curves or bounded curves, the endpoints of which coincide with contraction points or lie on identification curves; see Figure 1. The "abstract" *HasBoundaryTraits* sub-hierarchy lists additional predicates required to handle both curves that reach or approach the boundaries of the parameter space. It has no models. The refined *BoundedBoundaryTraits* and *UnboundedBoundaryTraits* sub-hierarchies list additional predicates required to handle bounded and unbounded curves respectively. The geometry-traits class that handles arcs of great circles models the *BoundedBoundaryTraits* concept, as the parameter space is bounded in all four directions. Finally, the *AllBoundaryTraits* sub-hierarchy refines all the above. A model of this concept can handle unbounded curves in some directions and bounded curves in others.

3 Handling arcs of great circles on the sphere

We use the following parameterization of the unit sphere: $\mathbb{P} = [-\pi, \pi] \times [-\frac{\pi}{2}, \frac{\pi}{2}]$ and $f_S(u,v) = (\cos u \cos v, \sin u \cos v, \sin v)$. This parameterization induces two contraction points $p_s = (0,0,-1)$ and $p_n = (0,0,1)$, referred to as the south and north poles respectively, and an identification curve that coincides with the opposite Prime (Greenwich) Meridian.

The geometry-traits class for geodesic arcs on the sphere is parameterized with a geometric kernel [10] that encapsulates the number type used to represent coordinates of geometric objects and to carry out algebraic operations on those objects. The implementation handles all degeneracies, and is exact as long as the underlying number type supports the arithmetic operations $+$, $-$, $*$, and / in unlimited precision over the rationals, such as the one provided by GMP[3]. A point in our arrangement is defined to be an unnormalized vector that emanates from the origin, extended with an enumeration that indicates whether the vector (i) pierces the south pole, (ii) pierces the

[1] `http://www.cgal.org`

[2] We do not support surfaces, which contain a contracted identification curve.

[3] `http://www.swox.com/gmp/`

north pole, (iii) intersects the identification arc, or (iv) is in any other direction. An arc of a great circle is represented by its two endpoints, by the normal of the plane that contains the arc, and some Boolean flags that cache information. The orientation of the plane and the source and target endpoints determine which one of the two great arcs is considered. The flags are used to expedite the performance.

All the required geometric operations listed in the traits concept are implemented using only rational arithmetic. Degeneracies, such as overlapping arcs that occur during intersection computation, are properly handled. The end result is a robust yet efficient implementation.

4 Applications

Armed with the geometry-traits for geodesic arcs on the sphere, we can use all the arrangement machinery to solve a variety of problems involving such arrangements. In particular, we compute Minkowski sums of convex polyhedra [7], by overlaying their respective Gaussian maps, which are arrangements of geodesics on the sphere. We also compute various Voronoi diagrams on the sphere through the computation of the lower envelope of the site-distance functions over the sphere. This section describes the latter application.

We define lower envelopes of functions on the sphere in a way similar to the standard definition of lower envelopes of bivariate functions in space [8]:

Definition 1 *Given a set of bivariate functions* $F = \{f_1, \ldots, f_n\}$*, where* $f_i : \mathbb{S}^2 \to \mathbb{R}$*, their* lower envelope $\Psi(u, v)$ *is defined to be their pointwise minimum* $\Psi(u, v) = \min_{1 \le i \le n} f_i(u, v)$.

The *minimization diagram* $\mathcal{M}(F)$ of the set F is the two-dimensional map obtained by central projection of the lower envelope onto $\mathbb{S}^2$.

Definition 2 *Given two points* $p_i, p_j \in \mathbb{S}^2$*, the* distance *between them* $\rho(p_i, p_j)$ *is defined to be the length of a geodesic arc that connects* p_i *and* p_j.

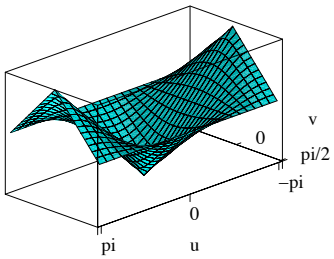
Definition 3 *Given a set of* n *points* $P = \{p_1, \ldots, p_n\}$*,* $p_i \in \mathbb{S}^2$*, we define* $R(P, p_i) = \{x \in \mathbb{S}^2 \mid \rho(x, p_i) < \rho(x, p_j), j \neq i\}$*.* $R(P, p_i)$ *is the region of all points that are closer to* p_i *then to any other point in* P.

The *Voronoi diagram* of P over $\mathbb{S}^2$ is defined to be the regions $R(P, p_1), R(P, p_2), \ldots, R(P, p_n)$ and their boundaries.

Edelsbrunner and Seidel [6] observed the connection between Voronoi diagrams in $\mathbb{R}^d$ and lower envelopes of the corresponding distance functions to the sites in $\mathbb{R}^{d+1}$. This also holds for our spherical case. From the above definitions it is clear that if $f_i : \mathbb{S}^2 \to \mathbb{R}$ is set to be $f_i(x) = \rho(x, p_i)$, for $i = 1, \ldots, n$, then the minimization diagram of $\{f_1, \ldots, f_n\}$ over $\mathbb{S}^2$ is exactly the Voronoi diagram of P over $\mathbb{S}^2$.

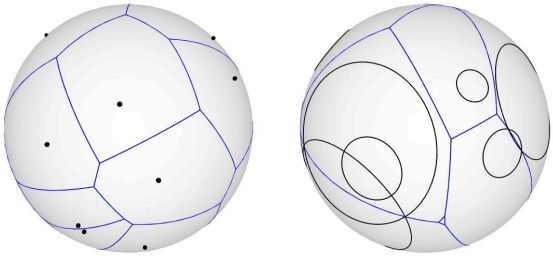
A new framework based on the envelope algorithm of CGAL [13] was developed to compute different types of Voronoi diagrams. The implementation is exact and can handle degenerate input. The framework provides a reduced and convenient interface between the construction of the diagrams and the construction of envelopes, which in turn are computed using the `Envelope_3` package [14]. Obtaining a new type of Voronoi diagrams only amounts to the provision of a traits class that handles the type of bisector curves of the new diagram type [9]. This traits class models the *EnvelopeVoronoiTraits* concept that refines one of the traits concepts mentioned in Section 2. Essentially, every type of Voronoi diagram, the bisectors of which can be handled by an arrangement traits class, can be implemented using this framework. The bisector curves between point sites on the sphere are great circles [16, 17], handled by the newly developed traits class described in Section 3; see Figure 2(a).

We implicitly construct envelopes of distance functions defined over the sphere to compute Voronoi diagrams. The image to the right illustrates the distance function from $(0, 0) \in [-\pi, \pi] \times [-\frac{\pi}{2}, \frac{\pi}{2}]$ on the sphere in the parameter space. The great circle bisector of two point sites on the sphere is the intersection of the sphere and the bisector plane of the points in $\mathbb{R}^3$ (imposed by the Euclidean metric).



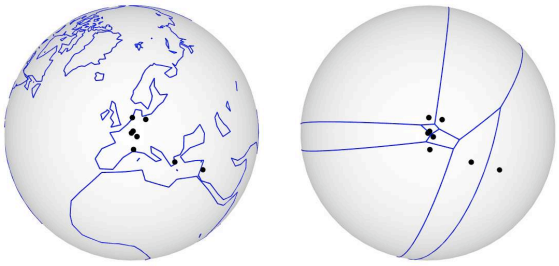
The envelope code together with the traits class for geodesic arcs on the sphere enable the computation of Voronoi diagrams on the sphere, the bisectors of which are great circles or piecewise curves composed of geodesic arcs. Another type of Voronoi diagrams whose bisectors are great circles is the power diagram of circles on the sphere [18], which generalizes the Voronoi diagram of points; see Figure 2(b). Power diagrams on the sphere have several applications similar to the applications of power diagrams in the plane. For example, determining whether a point is included in the union of circles on the sphere, and finding the boundary of the union of circles on the sphere [11, 18].

Given two circles on the sphere c_1 and c_2, let p_1 and p_2 be the planes containing c_1 and c_2 respectively. The bisector of c_1 and c_2 is the intersection of the sphere and the plane that contains the intersection line of p_1 and p_2 and the origin. If p_1 and p_2 are parallel planes, then the bisector is the intersection of the sphere and the plane that contains the origin and is parallel to both p_1 and p_2.



(a) (b)

Figure 2: Voronoi diagrams on the sphere. Sites are drawn in black and Voronoi edges are drawn in blue. (a) A Voronoi diagram of 14 random points. (b) A power diagram of 10 random circles.



(a) (b)

Figure 3: Arrangements on the sphere.

Figure 3(a) shows an arrangement on the sphere induced by (i) the continents and some of the islands on earth, and (ii) the institutions that participate in the ACS project,[4] which appear as isolated vertices. The sphere is oriented such that Nancy is at the center. The arrangement consists of 1053 vertices, 1081 edges, and 117 faces. The data was taken from gnuplot[5] and from google maps[6]. Figure 3(b) shows an arrangement that represents the Voronoi diagram of the eight cities, the institutions above are located at, namely Athens, Berlin, Groningen, Nancy, Saarbrücken, Sophia-Antipolis, Tel Aviv, and Zurich. The figure above shows the *overlay* of the two arrangements shown in Figure 3. Recall that arrangement points are represented as an unnormalized vector; see Section 3. The coordinates of such points are converted into machine floating-point only for rendering purposes.

References

[1] F. Aurenhammer. Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345–405, 1991.

[2] F. Aurenhammer and R. Klein. Voronoi diagrams. In J. Sack and G. Urrutia, editors, *Handb. Comput. Geom.*, chapter 5, pages 201–290. Elsevier, 2000.

[4] Algorithms for Complex Shapes: `http://acs.cs.rug.nl`

[5] `http://www.gnuplot.info/`

[6] `http://maps.google.com/`

[3] E. Berberich, E. Fogel, D. Halperin, K. Melhorn, and R. Wein. Sweeping and maintaining two-dimensional arrangements on surfaces: A first step. In *Proc. 15th Annu. Eur. Symp. Alg.*, pages 645–656, 2007.

[4] E. Berberich and M. Kerber. Exact arrangements on tori and Dupin cyclides, 2008. Manuscript.

[5] F. Cazals and S. Loriot. Computing the exact arrangement of circles on a sphere, with applications in structural biology. Technical Report 6049, Inria Sophia-Antipolis, 2006.

[6] H. Edelsbrunner and R. Seidel. Voronoi diagrams and arrangements. *Disc. Comput. Geom.*, 1:25–44, 1986.

[7] E. Fogel and D. Halperin. Exact and efficient construction of Minkowski sums of convex polyhedra with applications. *Computer-Aided Design*, 39(11):929–940, 2007.

[8] D. Halperin. Arrangements. In J. E. Goodman and J. O'Rourke, editors, *Handb. Disc. Comput. Geom.*, chapter 24, pages 529–562. Chapman & Hall/CRC, 2nd edition, 2004.

[9] D. Halperin, O. Setter, and M. Sharir. Exact and efficient construction of general two-dimensional Voronoi diagrams via divide and conquer of envelopes in space, 2008. Manuscript.

[10] S. Hert, M. Hoffmann, L. Kettner, S. Pion, and M. Seel. An adaptable and extensible geometry kernel. In *Proc. Workshop Alg. Eng.*, volume 2141 of *LNCS*, pages 79–90. Springer, 2001.

[11] H. Imai, M. Iri, and K. Murota. Voronoi diagram in the Laguerre geometry and its applications. *SIAM J. on Computing*, 14(1):93–105, 1985.

[12] R. Kunze, F. Wolter, and T. Rausch. Geodesic Voronoi diagrams on parametric surfaces. In *Computer Graphics Int. Conf.*, page 230, Washington, DC, USA, 1997. IEEE Computer Society.

[13] M. Meyerovitch. Robust, generic and efficient construction of envelopes of surfaces in three-dimensional space. In *Proc. 14th Annu. Eur. Symp. Alg.*, pages 792–803, 2006.

[14] M. Meyerovitch, R. Wein, and B. Zukerman. 3D envelopes. In Cgal Editorial Board, editor, Cgal *User and Reference Manual*. 3.3 edition, 2007.

[15] R. E. Miles. Random points, sets and tessellations on the surface of a sphere. *The Indian J. of Statistics*, 33:145–174, 1971.

[16] H.-S. Na, C.-N. Lee, and O. Cheong. Voronoi diagrams on the sphere. *Comput. Geom. Theory Appl.*, 23(2):183–194, 2002.

[17] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, NYC, 2nd edition, 2000.

[18] K. Sugihara. Laguerre Voronoi diagram on the sphere. *J. for Geom. Graphics*, 6(1):69–81, 2002.

[19] R. Wein, E. Fogel, B. Zukerman, and D. Halperin. 2D arrangements. In Cgal Editorial Board, editor, Cgal *User and Reference Manual*. 3.3 edition, 2007.

Voronoi Diagram of Ellipses in CGAL

Ioannis Z. Emiris* Elias P. Tsigaridas† George M. Tzoumas‡

Abstract

We present a CGAL-based implementation of the Voronoi diagram of ellipses in 2D. Based on the package for the Apollonius diagram in the plane and exploiting the generic programming principle, our main additions concern the implementation of the predicates. For this, we develop practical algebraic methods like trivariate system resultant computation, thus illustrating the concept of algebraic support to a geometric library via an algebraic kernel.

1 Introduction

This paper discusses our implementation of an algorithm for the Euclidean Voronoi diagram of ellipses, in the exact computation paradigm, cf. fig. 1. This is the first complete solution of how to implement the Voronoi diagram (and Delaunay graph) of ellipses under the exact computation paradigm, a non-trivial problem tackled in nonlinear computational geometry, because of the complexity of the algebraic operations involved. In particular, the real algebraic numbers involved in the InCircle predicate (defined later) are of degree 184.

We apply the incremental algorithm of [6] where the insertion of a new ellipse to the current Voronoi diagram consists of the following: (i) Find a conflict between an edge of the current diagram and the new ellipse, or detect that the latter is internal (hidden) in another ellipse, in which case it does not affect the diagram. (ii) Find the entire conflict region of the new ellipse and update the dual Delaunay graph.

We focus on *non-intersecting* ellipses, given *parametrically* (or *constructively*) in terms of their axes, center and rotation angle, which are all rational [3]. Our code is based on the CGAL package for the Apollonius (or Voronoi) diagram of circles [2], which uses the same incremental algorithm. CGAL follows the generic programming paradigm, hence the main issue was to analyze and implement all 4 predicates for ellipses: (κ_1) given two ellipses and a point, decide which ellipse is closest to the point; (κ_2) given two ellipses, decide the position of a third one relative to a specified external bitangent of the first two; (κ_3) given three ellipses, decide the position of a fourth one relative to one (external tritangent) Voronoi circle of the first three; this is the InCircle predicate; (κ_4) given four ellipses, compute the part of the Voronoi edge that changes due to the insertion of a new ellipse.

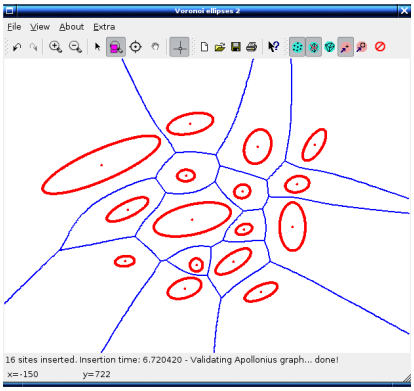


Figure 1: Voronoi diagram of ellipses.

The theoretical foundations of our approach were laid in [3]. A certified iterative subdivision algorithm with quadratic convergence was proposed in [4] for answering the InCircle predicate fast in non-degenerate instances using interval arithmetic and was a significant improvement over the corresponding algorithm in [3]. In [4], the authors worked with bisectors in the parametric space. So does the approach of [5] which is more general, since it applies to arbitrary Bézier or B-spline curves. However, this goes up to machine precision and is slower on ellipses.

Our main contribution is the first implementation[1] for the exact Voronoi diagram of ellipses, including a Graphical User Interface (GUI) for input, and visualization tools that can be used for any parametric curve. Our methods generalize to intersecting ellipses and pseudo-circles, but also to arbitrary closed smooth parametric curves. They can also compute an approximate Voronoi diagram of ellipses, with any predetermined precision. We implement certain algebraic methods in C++ to support InCircle, thus improving upon previous implementations. State-of-the-art *general* solvers are slow on InCircle, compared to our adapted solutions [3, 4].

The implemented algebraic methods rely, in specific

*University of Athens, Greece, emiris@di.uoa.gr
†INRIA Sophia-Antipolis, France, elias.tsigaridas@inria.fr
‡University of Athens, Greece, geotz@di.uoa.gr

[1] www.di.uoa.gr/~geotz/vorell/

ways, on the algebraic library SYNAPS [2] and, through this software, library MPFR,[3] and NTL.[4] We thus illustrate the concept of providing algebraic support to a geometric computing library such as CGAL. Our algorithms are described in sec. 2.

Lastly, in sec. 4 we report on experiments when we vary the number of ellipses, their bitsize, and the degeneracy of their configuration. We also test our code on sets of circles, which allows us to compare it against the Apollonius package; our software is one to two orders of magnitude slower, which is to be expected because of the adapted methods employed for circles.

2 The InCircle predicate

An ellipse is given in rational parametric form, which is suitable for following its boundary in the subdivision algorithm. Moreover, given the parametric form it is always possible to derive the implicit one using only rational arithmetic.

InCircle is clearly the most challenging predicate and the only one that had not been satisfactorily implemented in C++. In order to decide InCircle, we need to represent the Voronoi circle suitably. In [3], we proved that there can be 184 complex tritangent circles to three ellipses, hence this is the degree of the algebraic numbers involved in an algebraic approach. The corresponding algebraic system is very costly to solve, see [3], therefore we take a different track. First, we apply the certified subdivision solver of [4] and, if the predicate cannot be decided, we solve system $\{Q, B_1, B_3\}$ below. This happens at (near)degenerate configurations, where any fixed precision may not suffice.

We express the Voronoi circle by considering the intersection of three bisectors, namely the system: $B_1(t,r) = B_2(r,s) = B_3(s,t) = 0$, where t, r, s are the parameters of the three ellipses. This system is used in [4] for the subdivision algorithm with quadratic convergence. Experiments showed that it is very expensive to compute its resultant in order to use it for exact solving, due to the degrees of the B_i's.

We employ resultants, which expresses the solvability of a system of $n+1$ equations in n variables, as a condition on the coefficients. For the required notions from computer algebra, see [8].

Consider the following alternative polynomial system: $Q(t,r,s) = B_1(t,r) = B_3(s,t) = 0$. Here, Q is the condition that makes the three normals of each ellipse intersect at a single point. Q is a polynomial of total degree 12, four in each variable t, r, s. This system has a mixed volume of 432, like the system of $\{B_1, B_2, B_3\}$ above, but, nonetheless, it leads to an efficient way of computing the resultant. In fact, by exploiting the fact that not all variables appear in all equations, we can compute the system's resultant via two Sylvester resultants. The resultant that eliminates s from two polynomials is denoted by res_s.

$$\begin{aligned} R_1(t,r) &= \mathrm{res}_s(Q(t,r,s), B_3(s,t)) \\ &= \underbrace{(at^{28}r^{24} + \cdots)}_{\bar{R}_1} P_1(t)(1+t^2)^4, \\ R_2(t) &= \mathrm{res}_r(\bar{R}_1(t,r), B_1(t,r)) \\ &= R(t)[P_2(t)]^6(1+t^2)^{28} \qquad (1) \end{aligned}$$

where P_1, P_2 are univariate polynomials of degree 12, $\bar{R}_1$ is the "interesting" factor of R_1 and R is a univariate polynomial of degree 184 which is the polynomial we are looking for.

Lemma 1 *The factorization of $R_1(t,r)$, given in expression (1), is always true. The shown factors of $R_2(t)$ are always present; if they appear at the indicated powers, then expression (1) gives the full factorization of $R_2(t)$.*

The proof is omitted, but can be found in [1]. P_1 and P_2 correspond to the condition that the normals to two ellipses are parallel. The last factor in each factorization has no real roots.

We have no complete proof for the exponents of the extraneous factors in $R_2(t)$. Still, the shown exponents are confirmed by every example we have tried. In practice, we divide out these factors until we obtain the resultant with the optimal degree, namely 184.

The above lemma and discussion allow us to exploit the fact that both Sylvester determinants are factored. The factors which have no real roots, or whose roots correspond to the normals being collinear (which is a case handled apart), are divided out. Hence, our approach reduces to solving a univariate polynomial over the reals and comparing real algebraic numbers; the latter may require a univariate GCD computation to identify the case of common roots. To the best of our knowledge, there is no C++ implementation capable of computing efficiently the resultant that appears in InCircle.

The certified subdivision of [4] exploits several geometric properties of the problem, and allows us to decide InCircle before full precision has been reached. The algorithm approximates Voronoi circle's tangency points with precision up to 10^{-15} in up to 100 msec, when using standard floating-point arithmetic. When the specified precision is not enough (in near-degenerate or degenerate cases), we proceed with the resultant computation.

3 Implementation

For the implementation of the required algebraic operations, we have relied on algebraic library SYNAPS

[2] synaps.inria.fr/
[3] www.mpfr.org/
[4] www.shoup.net/ntl/

and, to be more precise, to library Mathemagix.[5] First, we implemented Newton's iteration with interval arithmetic as part of the `subdivix` package of Mathemagix (or SYNAPS). This is important for the numerical iterative algorithm for INCIRCLE. The floating-point types, and the associated arithmetic operations, are implemented in the MPFR library.

In order to solve the degree-184 univariate polynomials for INCIRCLE, we use the fast implementation of Continued Fractions [7], available in SYNAPS. This is comparable and, in most cases, faster than state-of-the-art exact univariate solvers such as RS, but also numeric solvers such as MPSOLVE.[6] The algorithm isolates all real roots in rational intervals, which also allows us to compare roots from different resultants.

We implemented bivariate and univariate polynomial interpolation in SYNAPS, in order to compute the resultant of the system of bisectors, refer to eq. (1). This is essentially a Chinese remaindering algorithm in the ring of polynomials, where the moduli are values of the polynomial and the output are its coefficients. After computing the numeric Sylvester determinants by NTL (see below), we use lemma 1 to divide out the values of the factors described in that lemma. Therefore, the number of moduli does not depend on the total degree of the determinant but rather on the total degree of the factor of interest, considered as a univariate or bivariate polynomial.

When two roots are equal, we compute the GCD of the defining polynomials. For this, we use NTL which is, to the best of our knowledge, the only open source C++ library that provides efficient implementation of asymptotically fast algorithms for polynomial GCD's and univariate Sylvester-resultant computation. We developed an interface between NTL and SYNAPS. The interface eliminates the need for unnecessary copies, and allows us to work directly with NTL objects in SYNAPS (and vice versa). We take advantage of the fact that integer arithmetic in NTL can be based on GMP, and we built a wrapper around these objects.

The concept of an `Algebraic_kernel` for the Voronoi diagram of ellipses is quite demanding. It requires, just to name the most important operations, symbolic univariate polynomial resultant and GCD computations, real root isolation and comparison of real algebraic number and computation of the resultant of a trivariate polynomial system.

In connecting to algebraic software, it was important to separate the geometric from the algebraic operations. This work is in line with the concept of kernels, adopted by CGAL, where different types of operations are grouped into separate modules.

For the combinatorial part of the algorithm, we relied on CGAL. CGAL follows the generic programming paradigm, hence the main issue is to implement the 4 predicates for ellipses and generalize circular sites to ellipses. We are based on the `Apollonius_graph_2` package. Besides the predicates, the most important class is the `Apollonius_site_2`, which represents the sites of the Voronoi diagram; in our case the ellipses. We modified the corresponding class so as to inherit from our `Ellipse` class. A snapshot of the corresponding code is as follows:

```
template < class K >
class Apollonius_site_2: public Ellipse
{
public:
  typedef K                          Kernel;
  typedef typename K::Point_2        Point_2;
  typedef Apollonius_site_2<K>       Self;
  typedef typename K::FT             FT;
// Field Number Type
  typedef typename K::RT             RT;
// Ring Number Type
.... };
```

4 Experiments

With elliptic sites. Let us now see several results regarding elliptic sites. These experiments have been carried out on a P4 2.6-GHz machine with 1.5GB of RAM.

First, we consider the overall time for the construction of the combinatorial structure of the dual (Delaunay) graph. While the first few sites are inserted almost instantly, subsequent ones cause many updates on the graph and require about a second for their insertion. Total timings are shown in the first two columns of table 1 (top). The runtime increases roughly linearly with 15 sites or more and is about one second for each ellipse. This is in accordance with theoretical bounds, although larger instances shall have to be tested to verify this.

We have also measured the performance of the first three predicates with varying bitsize (cf. the subsequent columns of the table). Using randomly perturbed coefficients by either adding or subtracting 10^{-e}, we obtain large rational numbers. All runtimes appear to grow subquadratically in e. For predicates κ_1 and κ_2 the timings vary from a couple milliseconds for $e = 20$ to 280 msec for $e = 100$. For INCIRCLE, the subdivision algorithm was used, since the situation was non-degenerate, and was roughly 10 times slower. In case of degeneracies, the runtime of INCIRCLE is dominated by the resultant computation, varying from 25 sec for $e = 4$ to 231 sec for $e = 20$. Note that the resultant computation is over 50 times slower than the time required by the subdivision algorithm.

Finally, we measured the time needed for the subdivision algorithm to reach a precision of 2^{-b} when using MPFR floats. This version currently lacks some

[5] www.mathemagix.org/

[6] www.dm.unipi.it/cluster-pages/mpsolve/

# sites	insertion	e	κ_1	κ_2	κ_3
5	0.752	20	0.0270	0.033	0.49
10	5.916	40	0.0696	0.076	1.02
15	13.000	60	0.1286	0.135	1.70
20	19.200	80	0.1996	0.197	2.49
25	25.602	100	0.2848	0.285	3.40

e	resultant	b	subdivision
4	25.12	53	0.20
8	57.24	900	0.58
12	102.63	2000	1.06
16	160.44	12000	10.65
20	231.19	24000	31.61

Table 1: Performance of the predicates on elliptic sites. Timings are in seconds.

optimizations making it about 2 times slower than the one in [4] using ALIAS.[7] Standard floating-point precision is achieved in about 0.2s, while 1 second suffices for almost 2000 bits of precision. However, higher approximations slow down considerably, such as the 24k-bit approximation that needs about half a minute. This raises some questions on whether the theoretical separation bound of several million bits can be achieved in practice by any implementation.

Against the CGAL Apollonius package. We performed experiments per predicate, when the input is restricted to circles. We used three data sets, involving degenerate inputs, near-degenerate inputs (by randomly perturbing the degenerate ones by 10^{-e}) and random inputs. These experiments were carried out on a 1.83GHz Core 2 Duo processor with 1GB of RAM. The timings for the Apollonius package were several milliseconds, while for our implementation varied from several milliseconds to a few seconds.

In most cases, degenerate inputs are solved faster than near-degenerate ones because the former, from the algebraic point of view, imply lower degree algebraic numbers. This is not the case for the subdivision-based algorithm, since in the degenerate cases it has to use large precision in order to make a decision. Another interesting observation is that runtime increases linearly in the bitsize, especially with perturbed input. This is because we implemented algorithms of constant arithmetic time complexity, which do not depend on how close to degeneracy the configuration lies.

It seems that our current implementation of the predicates for ellipses is up to two orders of magnitude slower that the dedicated one, when we restrict to circles. The worst relative performance is observed, as expected, for INCIRCLE, which is the most expensive predicate and, in the case of circles, has been optimized. The best relative performance occurs for κ_2, because the two approaches follow similar algorithms. The difference of performance is not surprising, since the case of circles reduces to computations with real algebraic numbers of degree 2 and the Apollonius predicates are specifically designed to exploit this.

We may conclude that specialized implementations for predicates involving small degree algebraic numbers, combined with algorithms exploiting the geometric characteristics of the problem, are more efficient than generic approaches.

[7] www-sop.inria.fr/coprin/logiciels/ALIAS/

Acknowledgments. George Tzoumas is partially supported by State Scholarship Foundation of Greece, Grant No. 4631. All authors acknowledge partial support by IST Programme of the EU as a Shared-cost RTD (FET Open) Project under Contract No IST-006413-2 (ACS - Algorithms for Complex Shapes). We thank Athanasios Kakargias for help with the overall code design, Michael Hemmer for suggestions on kernel design, and Costas Tsirogiannis for help with the GUI.

References

[1] I. Z. Emiris, E. P. Tsigaridas, and G. M. Tzoumas. A CGAL-based implementation for the Voronoi diagram of ellipses. Submitted. Manuscript available from http://www.di.uoa.gr/~geotz/.

[2] I.Z. Emiris and M.I. Karavelas. The predicates of the Apollonius diagram: algorithmic analysis and implementation. *Comp. Geom.: Theory & Appl., Spec. Issue on Robust Geometric Algorithms and their Implementations*, 33(1-2):18–57, 2006.

[3] I.Z. Emiris, E.P. Tsigaridas, and G.M. Tzoumas. The predicates for the Voronoi diagram of ellipses. In *Proc. Annual ACM Symp. on Computational Geometry*, pages 227–236, June 2006.

[4] I.Z. Emiris and G.M. Tzoumas. A real-time implementation of the predicates for the Voronoi diagram of parametric ellipses. In *Proc. ACM Symp. Solid Physical Modeling*, pages 133–142, Beijing, 2007.

[5] I. Hanniel, R. Muthuganapathy, G. Elber, and M.-S. Kim. Precise Voronoi cell extraction of free-form planar piecewise c^1-continuous closed rational curves. In *Proc. ACM Symp. Solid Phys. Modeling*, pages 51–59, Cambridge, Mass., 2005. (Best paper award).

[6] M.I. Karavelas and M. Yvinec. Voronoi diagram of convex objects in the plane. In *Proc. Europ. Symp. Algorithms*, LNCS, pages 337–348. Springer, 2003.

[7] E.P. Tsigaridas and I.Z. Emiris. On the complexity of real root isolation using continued fractions. *Theor. Comp. Science, Spec. Issue Comput. Algebraic Geom. & Applications*, 392(1-3):158–173, February 2008.

[8] C.K. Yap. *Fundamental Problems of Algorithmic Algebra*. Oxford University Press, New York, 2000.

A CGAL-Based Univariate Algebraic Kernel and Application to Arrangements

Sylvain Lazard * Luis Peñaranda* Elias Tsigaridas †

Abstract

Solving univariate polynomials and multivariate polynomial systems is critical in geometric computing with curved objects. Moreover, the real roots need to be computed in a certified way in order to avoid possible inconsistency in geometric algorithms. We present a CGAL-based univariate algebraic kernel, which follows the CGAL specifications for univariate kernels. It provides certified real-root isolation of univariate polynomials with integer coefficients (based on the library RS) and standard functionalities such as basic arithmetic operations, gcd and square-free factorization.

We compare our implementation with that of other univariate algebraic kernels that follow the same CGAL specifications. In particular, we compare it to the one developed at MPII. We also apply this kernel to the computation of arrangements of univariate polynomial functions.

1 Introduction

Implementing geometric algorithms robustly is known to be a difficult task for two main reasons. First, all degenerate situations have to be handled and second, algorithms often assume a real-RAM model which is not realistic in practice. In recent years, the paradigm of exact geometric computing arose as a standard for robust implementations. In this paradigm, geometric decisions, such as "is a point inside, outside or on a circle?", are made exactly, usually using exact arithmetic combined with interval arithmetic for efficiency; on the other hand, geometric constructions, such as the coordinates of a point of intersection, may be approximated.

We address here one recurrent difficulty arising when implementing algorithms dealing with curved objects. Such algorithms usually require evaluating, manipulating and solving systems of polynomials equations and comparing their roots. One of the most critical parts of dealing with polynomials or polynomial systems is the isolation of the real roots and their comparison.

We restrict here our attention to the case of univariate polynomials and address this problem in the context of CGAL, a C++ Computational Geometry Algorithms Library, which is an open source project and became a standard for the implementation of geometric algorithms [3].

CGAL is designed in a modular fashion. Algorithms are typically parameterized by a *traits* class which encapsulates the geometric objects, predicates and constructions used by the algorithm. Typically, this allows implementing algorithms independently of the type of input objects. For instance, a sweep-line algorithm for computing arrangements can be implemented generically for segments or curves. Similarly, the model of computation, such as exact arbitrary-length integer arithmetic or approximate fixed-precision floating-point arithmetic are encapsulated in the concept of *kernel*. An implementation is thus typically separated in three layers, the geometric algorithm which relies on a traits class, which itself relies on a kernel for elementary operations. A choice of traits class and kernel gives freedom to the users and allows comparison.

We present here a kernel for solving and manipulating the real roots of univariate polynomials with integer coefficients which follows CGAL specifications [2]. In particular, this kernel performs the isolation and comparison of the real roots of such polynomials. The kernel also provides various operations on polynomials, such as gcd, which are central for manipulating algebraic numbers. We also present experimental results and compare our kernel with the one developed by Hemmer and Limbach [10].

2 Univariate algebraic kernel

We describe here our implementation of our univariate algebraic kernel. The two main requirements of the CGAL specifications, which we describe here, are the isolation of real roots and their comparison. We also describe our implementation of two important specific operations, greatest common divisor (gcd) computation and refinement of isolating intervals, that are needed, in particular, for comparing algebraic numbers.

Preliminaries. The kernel handles univariate polynomials and algebraic numbers. The polynomials have

*LORIA (INRIA, CNRS, Nancy Université) and INRIA Grand Est, Nancy, France. Firstname.Name(AT)loria.fr

†INRIA Méditerranée, Sophia-Antipolis, France. Elias.Tsigaridas(AT)inria.fr

integer coefficients and are represented by arrays of GMP arbitrary-length integers [9]. We implemented in the kernel the basic functions on polynomials, including basic arithmetics, evaluation, and input/output. An algebraic number that is a root of a polynomial F is represented by F and an isolating interval, that is an interval containing this root but no other. We implemented intervals using the MPFI library, which represents intervals with two MPFR arbitrary fixed-precision floating point numbers; note that MPFR is developed on top of the GMP library for multi-precision arithmetic [11] [12].

Root isolation. For isolating the real roots of univariate polynomials with integer coefficients, we developed an interface with the library RS [14]. This library is written in C and is based on Descartes' rule for isolating the real roots of univariate polynomials with integer coefficients.

We briefly detail here the general design of the RS library; see [13] for details. RS is based on an algorithm known as *interval Descartes* [4]; namely, the coefficients of the polynomials obtained by changes of variable, sending intervals $[a, b]$ onto $[0, +\infty]$, are only approximated using interval arithmetic when this is sufficient for determining their signs. Note that the order in which these transformations are performed in RS is important for memory consumption. The intervals and operations on them are handled by the MPFI library.

Another characteristic of RS is its memory management: it implements a *mark-and-sweep* garbage collector, which is well suited to RS needs.

Algebraic number comparison. As mentioned above, one of the main requirements of the CGAL algebraic kernel specifications is to compare two algebraic numbers r_1 and r_2. If we are lucky, their isolating intervals do not overlap and the comparison is straightforward. This is, of course, not always the case. If we knew that they were not equal, we could refine both isolating intervals until they do not overlap. See below for details on how we perform the refinements. Hence, the problem reduces to determining whether the algebraic numbers are equal or not.

To do so, we compute the gcd of the polynomials P_1 and P_2 associated to the algebraic numbers; see below for details on this operation. The roots of this gcd are the common roots of both polynomials. After calculating the gcd, we isolate its roots and refine the isolation intervals until each one of them overlap with exactly one root of P_1 and of P_2. If the isolating interval of r_1 and r_2 both overlap with the isolating interval of a root of the gcd, then $r_1 = r_2$. Otherwise they are not equal.

Gcd computations. Computing greatest common divisors between two polynomials is not a difficult task, however, it is not trivial to do so efficiently. Indeed, a naive implementation of the Euclidean algorithm works fine for small polynomials but the intermediate coefficients suffer an exponential grow in size, which is not manageable for medium to large size polynomials.

We thus implemented a *modular* gcd function, which calculates the gcd of polynomials modulo some prime numbers and reconstructs later the result with the help of the *Chinese remainder theorem*. Details on these algorithms can be found in [8]. Note modular gcd is always more efficient than regular gcd, especially when the two polynomials have no common roots.

Refining isolating intervals. As we mentioned before, refining the interval representing an algebraic number is critical for comparing such numbers. We have implemented two approaches for refinement.

Both approaches require that the polynomial associated to the algebraic number is square free. The first step thus consists of computing the square-free part of the polynomial. This is easily done by computing the gcd of the polynomial and its derivative.

Our first approach is a simple bisection algorithm. It consists in calculating the sign of the polynomial associated to the algebraic number at the endpoints and midpoint of the interval. Depending on those three signs we can take as isolating interval the left of right half of the previous one.[1]

The second approach we implemented is the *quadratic interval refinement* [1]. Roughly speaking, this method splits the interval in many parts and guesses in which one the root lies. If the guess is correct, the algorithm will divide, in the next refinement step the (chosen) interval in more parts and, if not, in less. Unfortunately, we can not always guarantee that we guess the correct interval at each step, so on average the algorithm turns out to be just a bit faster than the bisection one. Moreover, we have to implement its data structures very carefully in order to be efficient. In particular, this required the development of functions to handle dyadic numbers efficiently. Note that these functions are also useful in the bisection method when increasing the precision (because working directly with MPFR is rather tricky).

Currently, refinement function based on both approaches are present in our kernel and the user can choose the one best suited to her/his needs.

3 Benchmarks

In this section, we compare the running time for root isolation of our algebraic kernel with the one devel-

[1] Note that since the polynomial is square free the signs at the two endpoints of any isolating interval always differ. We thus do not need to compute the sign at both endpoints.

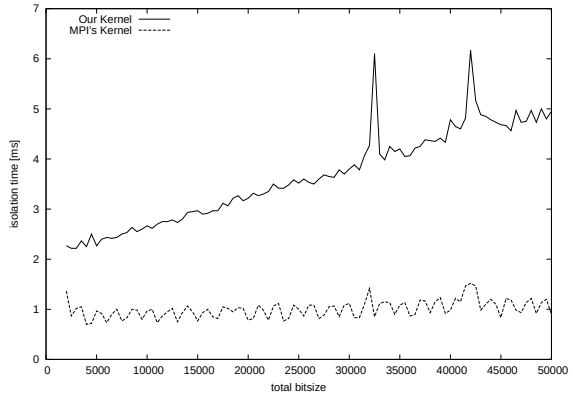


Figure 1: Degree 12 polynomials.

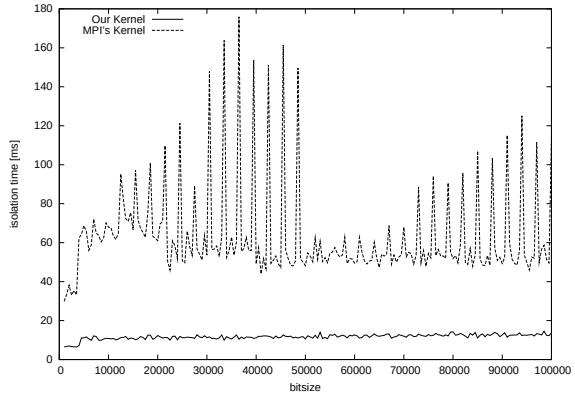


Figure 2: Polynomials of constant degree 100

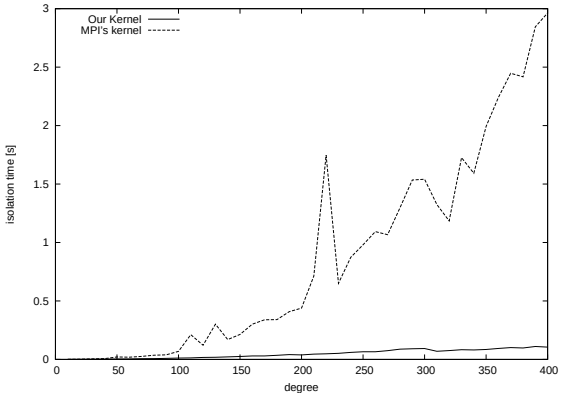


Figure 3: Polynomials of constant bitsize 20000

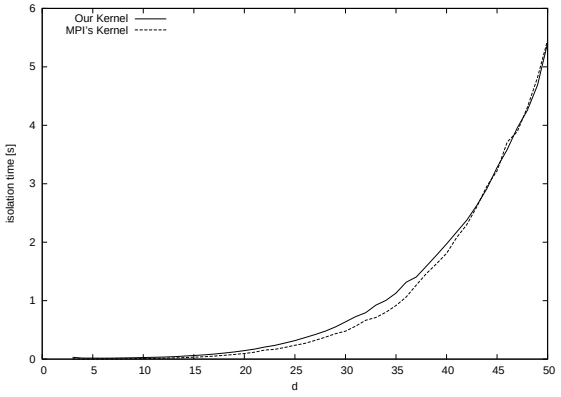


Figure 4: Mignotte polynomials

oped by Hemmer and Limbach [10]. Running times for root isolation is a very representative test because, as explained before, it involves other non-trivial operations on polynomials. For technical reasons, we could not compare our running time for comparing algebraic numbers with those of [10]. All tests were ran on a single-core 3.2 GHz Intel Pentium 4 with 2 Gb of RAM, using 64-bit Linux.

The first test sets comes from [10]. See Figure 1. It consists of polynomials of degree 12, each one being the product of six degree-two polynomials that have at least a root in the interval $[0, 1]$. The tests are averaged over 50 trials. The value represented on the x-axis is the total bitsize of all the coefficients of the input polynomial.

Secondly, we consider random polynomials with constant degree 100 and varying bitsize. See Figure 2. As before, the x-axis is the total bitsize of all the coefficients. The tests are averaged over 100 trials. We also consider random polynomials with constant bitsize 20000 but varying degrees. The results, shown in Figure 3 are averaged over 100 trials.

Finally, we tested Mignotte polynomials, that is nearly degenerate polynomials of the form $x^d - 2(kx - 1)^2$. The difficulty with solving these polynomials lies in the fact that two of their roots are very close to each other (the isolating intervals for these two roots are thus very small). For these tests, we used Mignotte polynomials with coefficients of bitsize 50, with varying degree d. Running time are shown in Figure 4: the tests are averaged over 5 trials.

4 Discussion

Figure 1 shows that our kernel's performance is worse than MPII's one for small degree polynomials. This difference comes from the fact that Rs, the most consuming part of our process, is conceived for handling high degree polynomials. This fact is confirmed by Figures 2 and 3, which show that for polynomials of larger degrees, our kernel runs faster.

We can also see in these two figures that the isolation time does not depend much on the bitsize of the input polynomials but mainly on the degree. This makes sense because of the considered algorithms for root isolation: *bitstream Descartes* [6] and *interval Descartes* [13] do not use, in most cases, all the bits of the coefficients. This should theoretically imply that, on random polynomials, the running time does not depend at all on the input bitsize. We however observe in Figure 1 that this is not quite the case for our kernel. This is presumably caused by the cost of copying the input polynomials to Rs memory space.

Despite this fact, we observe that the running time

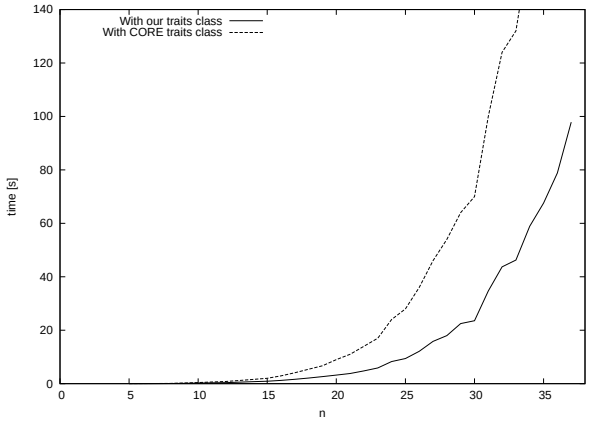


Figure 5: Arrangement calculation

of our implementation is very stable and does not depend too much on the bitsize or on the degree of the input polynomials. On the other hand, MPII's kernel does not either depend on the input bitsize. However it is much more dependent on the degree and is less stable.

Finally, we can mention that Figure 4 shows no significant difference between both kernels for isolating roots of Mignotte polynomials. MPII's kernel is barely faster for the smaller values of d, but that difference disappears when this value grows. However, this figure depicts the expected difficulty of solving Mignotte polynomials compared to random polynomials.

5 Arrangements

As an example of possible benefit of having efficient algebraic kernels in CGAL, we used our implementation to construct arrangements of polynomial functions. Wein and Fogel provided a CGAL package for calculating arrangements [15]. This package calculates the arrangements of general curves [7]. It is the user who must implement the data structures to store the curves and the primitive operations; requiring for example comparing positions of points, comparing the vertical order of curves at infinity and intersecting and splitting curves. All these functions must be grouped in a *traits class*, which is a transparent and convenient way to work with a package in CGAL. We implemented a traits class which uses the functions of our algebraic kernel and compared its performance with another traits classes which comes with CGAL's arrangement package and uses the CORE library [5].

To test the arrangement calculation, we generated n polynomials of degree $n-1$ with (n) coefficients of bitsize n. The running time for the construction of this type of arrangements is shown in Figure 5. We observe that we gain a factor of roughly two when using our kernel.

Acknowledgments

We would like to thanks M. Hemmer, E. Berberich, M. Kerber, and S. Limbach for fruitful discussion on the kernel developed at MPII and on the experiments.

References

[1] J. Abbott. *Quadratic interval refinement for real roots.* Poster in ISAAC, 2006.

[2] E. Berberich, M. Hemmer, M.I. Karavelas, and M. Teillaud. *Revision of the interface specification of algebraic kernel.* Technical Report, ACS-TR-243301-01, 2007.

[3] CGAL, Computational Geometry Algorithms Library. `http://www.cgal.org/`

[4] G.E. Collins, J.R. Johnson, and W. Krandick. *Interval arithmetic in Cylindrical Algebraic Decomposition.* Journal of Symbolic Computation, (2002) 34:145–157.

[5] CORE. `http://cs.nyu.edu/exact/`

[6] A. Eigenwillig, L. Kettner, W. Krandick, K. Mehlhorn, S. Schmitt and N. Wolpert. *A Descartes algorithm for polynomials with bit-stream coefficients.* In Proc. 8th Int. Workshop on Computer Algebra in Scient. Comput. (CASC), LNCS. Springer, 2005.

[7] E. Fogel, D. Halperin, L. Kettner, M. Teillaud, R. Wein and N. Wolpert. *Effective Computational Geometry for Curves and Surfaces. Chapter 1, Arrangements.* J.-D. Boissonnat. and M. Teillaud (editors), Springer, 2006.

[8] K.O. Geddes, S.R. Czapor and G. Labahn. *Algorithms for Computer Algebra.* Kluwer Academic Publishers, 1992.

[9] GMP. GNU multiple precision arithmetic library. `http://gmplib.org/`

[10] M. Hemmer and S. Limbach *Benchmarks on a generic univariate algebraic kernel* ACS Technical Report No. ACS-TR-243306-03, 2006.

[11] MPFI. Multiple precision interval arithmetic library. `http://perso.ens-lyon.fr/nathalie.revol/software.html`

[12] MPFR. Library for multiple-precision floating-point computations. `http://mpfr.org/`

[13] F. Rouillier and P. Zimmermann. *Efficient Isolation of Polynomial Real Roots.* Journal of Computational and Applied Mathematics, vol. 162 n. 1, p. 33-50, 2003.

[14] RS. Real roots of systems with a finite number of complex solutions. `http://fgbrs.lip6.fr/`

[15] R. Wein and E. Fogel *The new design of CGAL's arrangement package.* Technical report, Tel-Aviv University, 2005. `http://www.cs.tau.ac.il/~wein/publications/pdfs/Arr_new_design.pdf`

Generic Implementation of a Data Structure for 3D Regular Complexes

Antoine Bru * Monique Teillaud †

Abstract

We present the implementation of a data structure, based on and extending the CGAL Halfedge Data Structure, that allows to represent 3D manifold regular complexes.

1 Introduction

Whereas software for storing planar subdivisions are publicly available [2, 19, 12], as well as software for computing triangulations in 3D [18, 3, 16], software for storing more general 3D subdivisions are seldom found. Still, there is a strong need in practice for such data structures. Indeed, while algebraic issues are usually seen as the bottleneck for computing arrangements of quadric surfaces in 3D, an appropriate data structure is in fact also missing. Arrangements of the simplest quadrics, namely spheres, may have a complicated topology. However, when decomposed in a careful way, such as the so-called vertical decomposition [17, 15] the subdivision that is obtained is actually simply a regular complex.

We follow the definition of a *regular complex* given in [6], as the natural generalization of a simplicial complex to cells that are not necessarily simplices:

- A *k-cell* is homeomorphic to a closed ball of dimension k.
- A *regular complex* K is a finite collection of cells such that the following two conditions hold:
  - Cells have pairwise disjoint interiors.
  - The boundary of each cell c is the union of other cells in K.

We present in this paper the design and implementation of a cellular data structure capable of storing and traversing a manifold regular complex of dimension 3. We will use the standard terminology in 3D: vertices for 0-cells, edges for 1-cells, facets for 2-cells, and cells for 3-cells as long as no ambiguity arises.

In the restricted case of polyhedral subdivisions, our data structure can intuitively be seen as gluing together several polyhedral surfaces along their common facets, just as the CIEL structure is doing [13] (see Figure 1). The Nef polyhedra can store more general

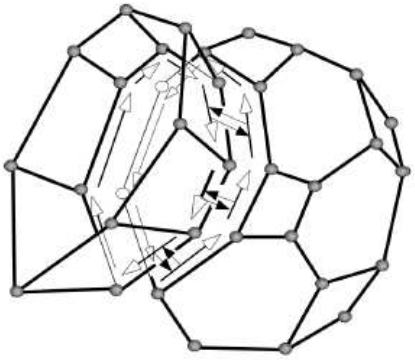


Figure 1: The CIEL data structure (picture from [13])

non-manifold structures but is intrinsically restricted to planar facets [4, 9]. Note that our definition does not imply that the facets be planar. The structure is independent from a geometric embedding into $\mathbb{R}^3$ but is flexible and generic enough to allow curved facets and edges.

The structure allows to represent a subdivision of a k-manifold in 3D. It is of course less general than the G-maps that represent subdivisions of quasi-manifolds in dimension n [14], and the cell-tuple structure that represent subdivided manifolds in dimension n [5]. For the special case of 3D simplicial complexes, a representation of non-manifold complexes is proposed in [10].

We show how the data structure can be implemented in the framework of the Computational Geometry Algorithms Library CGAL, following the generic programmic paradigm [1, 7, 8]. Our objective is, as more generally in CGAL, to develop flexible, efficient and easy-to-use software.

We also aim at reusing existing code as much as possible to avoid useless code duplication, which would be a waste, both in development and maintenance. More specifically, Section 2 shows how the principles of the CGAL Halfedge Data Structure (HDS for short) [12] are reused in our 3D Cellular Data Structure (CDS

*This work was done while the first author was appointed by INRIA Sophia Antipolis.

†INRIA Sophia Antipolis, Monique.Teillaud@sophia.inria.fr
http://www-sop.inria.fr/geometrica/team/

for short). Reusing the CGAL HDS prevents code duplication, it allows us to benefit from already existing functionalities, and allows our future users to see a uniform interface. However, it creates constraints in our implementation. Section 3 explains how difficulties are overcome, and hidden from the end user. Section 4 quickly mentions various functionality offered by the data structure.

2 Main principles

HDS. The halfedge data structure is a representation of a three-dimensional polyhedral surface [11]. In fact it can be used more generally to store non polyhedral two-dimensional manifold surfaces homeomorphic to spheres.

The HDS acts as a container of vertices, halfedges and faces, which are the *items* of the HDS. The sets of vertices, halfedges and faces are stored independently in either a doubly-connected list or a vector. Items are accessed through *handles* that can be roughly seen as smart pointers. The HDS offers functionality like insertion and removal of items, Euler operators, and iterators to traverse the structure.

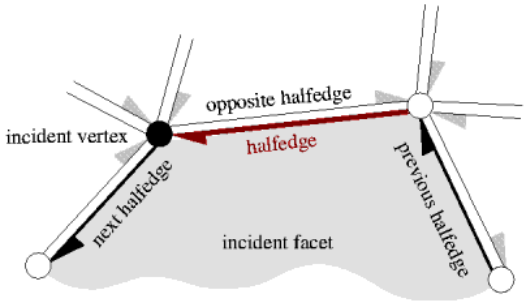


Figure 2: The halfedge data structure (picture from [12])

Each item manages its own adjacency and incidence relations (see Figure 2), which offers special flexibility. It is possible to activate or disable these relations according to the user need. For instance, a halfedge may store its incident face as a handle, added to the item structure. In this case, a `Face_handle` is added into the item structure `Halfedge`, and it can be accessed by the `face()` method as shown in Figure 4.

As the name says, HDS is a halfedge-based representation. Only halfedges are needed to describe a polyhedral surface as a graph. In the minimal situation when no option is activated, for each halfedge its next and opposite halfedges are available.

Items are easily extendable to add other informations like coordinates of vertices, or color into faces: The user can create new classes inherited from the item classes and declare them in wrapper structures. All these wrappers are gathered together in the same structure, called `Item`, which is given as a template parameter to the halfedge data structure. Figure 3 shows that the user extended items are used in the HDS items.

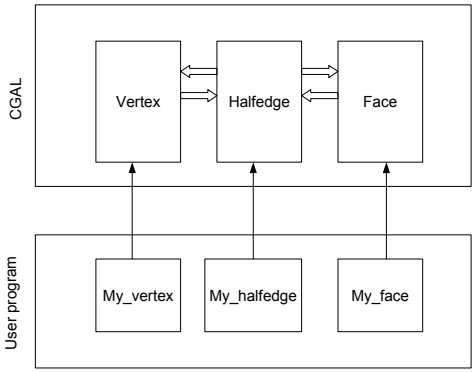


Figure 3: User defined items

```
template<class Refs, class Traits>
struct My_face
   : public CGAL::HalfedgeDS_face<Refs> {
 CGAL::color Color;
};

struct My_items
   : public CGAL::Items {
  template <class Refs, class Traits>
  struct Facet_wrapper {
    typedef My_face<Refs, Traits> Face;
  };
};

typedef CGAL::HalfedgeDS_list<My_items> HDS;
```

CDS. A regular complex is considered here like a set of 3D cells, that are polyhedra or more generally manifold surfaces, tightly glued together by their facets and edges (see Figure 1 for the polyhedral case). We represent a regular complex as a Cellular Data Structure (CDS), that can be seen as a container of k-cells for $k = 0, 1, 2, 3$.

As seen above, the HDS is based on the concept of halfedge. In a similar way, the CDS introduces the concept of two-dimensional *halffacet*. Each 2-cell (or facet) common to two 3-cells is split into two symmetric opposite halffacets belonging to the two cells incident to this facet.

The implementation of the CDS uses the flexibility and extensibility of the CGAL HDS, by modifying and reusing the HDS features.
◦ Each 3-cell (cell for short) is stored as a HDS.
◦ Halffacets reuse `Faces`. To this aim, an `opposite` field is added to the `Face` structure and points to another halffacet. This halffacet corresponds to the common facet of the two adjacent cells, seen from the

adjacent HDS.
∘ A given 1-cell (edge) of the regular complex corresponds to a full set of halfedges in the CDS, since an edge is incident to several facets in the complex. To complete the structure, each halfedge incident to a halffacet f of a cell c must store access to its `mirror` halfedge in the adjacent HDS: The `mirror` of e is the halfedge of the cell c' adjacent to c through f, that is incident to the halffacet f' opposite to f, and corresponds to the same edge of the regular complex

In the same way as the HDS, the CDS is a halfedge-based structure. Only halfedges, with their `next`, `opposite` and `mirror` halfedges, are necessary to describe entirely a regular complex.

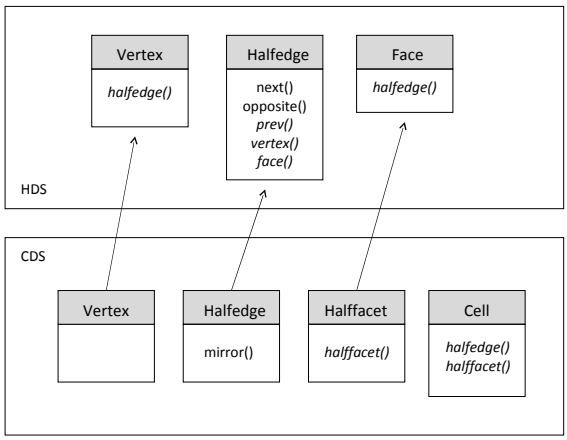


Figure 4: HDS and CDS items (optional fields are shown in italics)

3 Realization

As described above, a 3-cell is represented as a HDS, which is a container of k-cells for $k = 0, 1, 2$. However, a HDS stores 3 independent sets of faces, halfedges and vertices, so, a 2-cell (face) is not a container of halfedges and vertices. As a consequence, handling incidence and adjacency relations in the CDS is not straightforward, and we give in this section the main ideas that allow us to deal with this issue.

Cells. A new `Cell` item is added. It is templated by the HDS, which allows it to fetch the types for halfedge and halffacet from the HDS. In a symmetric way, the items vertices, halfedges and cells must now be templated by the CDS to fetch the cell type. A new implementation of these items, inherited from the HDS items, is provided.

HDS items are able to store other information than incidence and adjacency, for instance geometry or color. It is necessary to propose the same possibility for the cells of the CDS, and this is done in the same way as for other items: A cell wrapper structure is created for user customisations.

Vertices. In our representation of a regular complex, vertices do not have an important structural role, since CDS is, as HDS, a halfedge-based structure. The principal role of vertices is to carry information such as coordinates.

In the HDS implementation, the HDS holds its container of vertices. So, if HDS was used for representing cells in a naive way, a vertex incident to n cells would be stored as n independent objects, which we avoid in the following way.

A vertex container is added in the CDS. However, the sets (lists or vectors) of items of the HDS are declared into the class HDS and the handles, defined in Section 2, can't directly be references to sets outside the HDS. Still, the CDS can give its vertex handle type to each HDS through a minimum vertex item, that contains this new CDS vertex handle, and that is created and defined into the Vertex wrapper given as template parameter. In this way, a vertex of a HDS now becomes a vertex handle pointing on a vertex of the CDS.

CDS Items. To summarize, the CDS is templated by a structure containing the four items for vertex, halfedge, halffacet and cell. A second structure, `HDS_derived_item`, is used to propose the new vertex wrapper, and the halfedge, halffacet and cells wrappers. This structure is then used as a template of the HDS. See Figure 5.

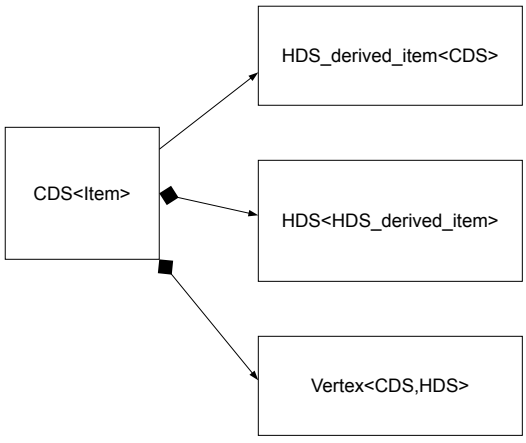


Figure 5: General design of the Cellular Data Structure

4 Functionality

We only roughly sketch functionality here.

As in the HDS, the CDS proposes various options allowing to have vertices, facets and cells, or only halfedges.

Reusing the HDS allows us to get all its iterators and circulators for free. Adding functionality like an iterator on the cell container, and using the mirror pointer of halfedges, we can then easily implement all iterators and circulators allowing to easily traverse the whole structure.

Acknowledgements. The authors wish to thank Laurent Rineau for helpful discussions.

References

[1] CGAL, Computational Geometry Algorithms Library. `http://www.cgal.org`.

[2] LEDA, Library for efficient data types and algorithms. `http://www.algorithmic-solutions.com/enleda.htm`.

[3] B. Barber *et al.* Qhull. `http://www.qhull.org`.

[4] H. Bieri and W. Nef. Elementary set operations with d-dimensional polyhedra. In *Computational Geometry and its Applications*, volume 333 of *Lecture Notes Comput. Sci.*, pages 97–112. Springer-Verlag, 1988.

[5] E. Brisson. Representing geometric structures in d dimensions: Topology and order. *Discrete Comput. Geom.*, 9:387–426, 1993.

[6] H. Edelsbrunner and N. R. Shah. Triangulating topological spaces. *Int. J. on Comp. Geom.*, 7:365–378, 1997.

[7] A. Fabri, G.-J. Giezeman, L. Kettner, S. Schirra, and S. Schönherr. On the design of CGAL a computational geometry algorithms library. *Softw. – Pract. Exp.*, 30(11):1167–1202, 2000.

[8] E. Fogel and M. Teillaud. Generic programming and the CGAL library. In J.-D. Boissonnat and M. Teillaud, editors, *Effective Computational Geometry for Curves and Surfaces*. Springer-Verlag, Mathematics and Visualization, 2006.

[9] P. Hachenberger and L. Kettner. 3d boolean operations on nef polyhedra. In C. E. Board, editor, *CGAL User and Reference Manual*. 3.3 edition, 2007.

[10] A. Hui, L. Vaczlavik, and L. De Floriani. A decomposition-based representation for 3D simplicial complexes. In *Proc. Eurographics Symposium on Geometry Processing*, pages 101–110, 2006.

[11] L. Kettner. Using generic programming for designing a data structure for polyhedral surfaces. *Comput. Geom. Theory Appl.*, 13:65–90, 1999.

[12] L. Kettner. Halfedge data structures. In C. E. Board, editor, *CGAL User and Reference Manual*. 3.3 edition, 2007.

[13] B. Lévy, G. Caumon, S. Conreaux, and X. Cavin. Circular incident edge lists: a data structure for rendering complex unstructured grids. In *Proc. IEEE Visualization*, pages 191–198, 2001.

[14] P. Lienhardt. N-dimensional generalized combinatorial maps and cellular quasi-manifolds. *Internat. J. Comput. Geom. Appl.*, 4(3):275–324, 1994.

[15] B. Mourrain, J.-P. Técourt, and M. Teillaud. On the computation of an arrangement of quadrics in 3d. *Computational Geometry: Theory and Applications*, 30:145–164, 2005. Special issue, 19th European Workshop on Computational Geometry.

[16] S. Pion and M. Teillaud. 3d triangulations. In C. E. Board, editor, *CGAL User and Reference Manual*. 3.3 edition, 2007.

[17] M. Sharir and P. K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications*. Cambridge University Press, New York, 1995.

[18] J. R. Shewchuk. Pyramid. `http://www.cs.cmu.edu/~quake/triangle.html`.

[19] R. Wein, E. Fogel, B. Zukerman, and D. Halperin. 2d arrangements. In C. E. Board, editor, *CGAL User and Reference Manual*. 3.3 edition, 2007.

Online Uniformity of Integer Points on a Line

Tetsuo Asano *

Abstract

This paper presents algorithms for computing a uniform sequence of n integer points in a given interval $[0, m]$, where uniformity of a point set is measured by the ratio of the minimum gap over the maximum gap. Several nontrivial results are shown.

1 Introduction

A number of applications need uniformly distributed points over a specific domain. It is commonly known that randomly generated points are not always good enough. In a mesh generation, for example, we have to distribute points uniformly over a region of interest to form good meshes. But, first of all, how can we measure the uniformity of points? In the theory of Discrepancy [3, 4] the uniformity of points is measured by how the number of points in a small region like an axis-parallel rectangle changes while moving around the domain, more formally by the difference (or discrepancy) between the largest and smallest numbers of points in the moving region. For normalization we usually divide the difference by the area of the moving region. Then, the discrepancy is given as the supremum of the ratios for all possible scales of the region. One of the difficulties here is hardness of such evaluation since we have to prepare all possible scales and all possible locations.

We consider a special case of such a problem, that is, how to insert n integer points in a given interval $[0, m]$ so that points are uniformly distributed, or in other words, the ratio of the minimum gap over the maximum gap is not so low. We present a simple algorithm for achieving the ratio $1/2$ for all integers m and n with $m > n > 0$. It is not trivial at all to achieve the ratio strictly greater than $1/2$. We characterize when n points can be inserted into an interval $[0, m]$ while keeping the ratio strictly greater than $1/2$.

2 Problem

An (n, m)-sequence is sequence of integers (or points of integral coordinates) $(p_1, \ldots, p_n)$ between 1 and $m-1$, i.e., p_i is an integer in the interval $[1, m-1]$ for $i = 1, \ldots, n$. For an (n, m)-sequence $(p_1, \ldots, p_n)$, we define a gap at p_i, denoted by $\delta(p_i)$, by

$$\delta^{[0,m]}(p_i) = \min\{|p_i - p_j|;\ j = 0, 1, \ldots, n, n+1, j \neq i\}, \quad (1)$$

where $p_0 = 0$ and $p_{n+1} = m$. Let $(p_0 = 0, p_{i_1}, \ldots, p_{i_n}, p_{n+1} = m)$ be a sorted list of the integers $p_0 = 0, p_1, \ldots, p_n, p_{n+1} = m)$. Then, for each integer p_{i_k} its associated gap is defined by

$$\delta^{[0,m]}(p_{i_k}) = \min\{|p_{i_k} - p_{i_{k-1}}|, |p_{i_k} - p_{i_{k+1}}|\}, \quad k = 1, \ldots, n. \quad (2)$$

The maximum and minimum gaps are denoted by $\delta^{[0,m]}_{\min}(p_1, \ldots, p_n)$ and $\delta_{\max}(p_1, \ldots, p_n)$, respectively.

It may be natural and reasonable to measure uniformity of a point set $\{p_0 = 0, p_1, \ldots, p_n, p_{n+1} = m\}$ by the ratio of the minimum and maximum gaps, that is, the (static) uniformity $\mu_s^{[0,m]}(p_1, \ldots, p_n)$ of the set is defined by

$$\mu_s^{[0,m]}(p_1, \ldots, p_n) = \frac{\delta_{\min}(p_1, \ldots, p_n)}{\delta_{\max}(p_1, \ldots, p_n)}. \quad (3)$$

In this paper we are interested in uniformity achieved by a sequence of points. That is, points are inserted one by one according to a given sequence. Every time when a point is inserted, we measure the uniformity of the point set. The worst uniformity we obtain before inserting all the points is defined to the online uniformity of the point sequence. Formally, we define the online uniformity $\mu^{[0,m]}(p_1, \ldots, p_n)$ for a point sequence $(p_1, \ldots, p_n)$ of length n in the interval $[0, m]$ by

$$\mu^{[0,m]}(p_1, \ldots, p_n) = \min_{k=1,\ldots,n} \{\mu_s^{[0,m]}(p_1, \ldots, p_k)\}. \quad (4)$$

An (n, m)-sequence is called *uniform* if its online uniformity is strictly greater than $1/2$.

3 Greedy algorithm

A natural idea is to halve the longest interval (maximum gap) to define a sequence of points on a line in a given interval. We can generalize this idea to higher dimensions. In higher dimensions we construct a Voronoi diagram for a set of points and choose a Voronoi vertex that is farthest from the closest point as the next point to insert. The performance of the greedy algorithm is not so bad. In fact, it achieves

*School of Information Science, JAIST, Japan, t-asano@jaist.ac.jp

the uniformity 1/2 in one dimension [1, 2]. However, it is not the case when points are limited to integer points. As a simple example consider a sequence of length 2 for an interval $[0, 6]$. The first point is 3, the midpoint of the interval. Then, we have two subintervals of length 3. Since we can only choose an integer point, one of the subinterval is divided into two subintervals of lengths 1 and 2. So, after choosing the two points the minimum gap is 1 while the maximum gap remains 3. So, the uniformity is $1/3 < 1/2$.

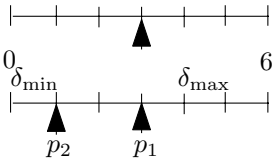


Figure 1: Behavior of Voronoi Insertion on integer points.

Another example is the case of $(m, n) = (10, 4)$. In the following we just maintain a set of lengths of intervals. Initially we have $\{10\}$. By the first point we must have $\{5, 5\}$ or $\{4, 6\}$. The even partition $\{5, 5\}$ does not lead to uniformity 1/2 since in the next division we have $\{2, 3, 5\}$, whose uniformity is $2/5 < 1/2$. So, $\{4, 6\}$ is the only choice and then we obtain the set $\{4, 2, 4\}$ by dividing the interval of length 6. Now, we can divide 4 into $\{2, 2\}$. Thus, the resulting set of interval lengths is $\{4, 2, 2, 2\}$ with uniformity $2/4 = 1/2$. On the other hand, if we divide 6 into $3, 3$, a bad partition, as we have seen before, leads to a bad situation. It is actually bad. After dividing 4 into $2, 2$ we have $\{2, 2, 3, 3\}$. We have to divide 3, but there is only one way of partition $3 \to \{1, 2\}$. Thus, the resulting set is $\{2, 2, 1, 2, 3\}$ with uniformity $1/3 < 1/2$. See Figure 2. This example implies that we should not take the midpoint even if there is a unique midpoint (note that there are two midpoints in an interval of odd length).

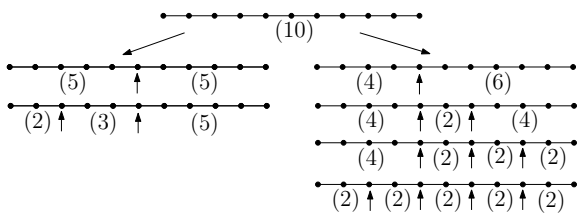


Figure 2: Partition of an interval of length 10 in two different ways. If we divide 10 into $5, 5$ as in the left figure, it is impossible to keep the uniformity $\geq 1/2$. The division $10 \to (4, 6)$ leads to a sequence with uniformity $\geq 1/2$.

Now, a natural question is whether there is an algorithm for finding a sequence of points with uniformity at least 1/2 for any pair of integers m and n with $m > n$. The following lemma answers the question in an affirmative way.

Lemma 1 *There is an algorithm for finding a sequence of points with uniformity at least* 1/2 *for any pair of integers* m *and* n *if* $m > n > 0$.

Proof. We prove the lemma in a constructive manner. The algorithm iteratively partitions the longest interval (maximum gap). An important thing is to divide an interval of length m into ones of lengths 2^k and the rest $r = m - 2^k$ using an integer k such that $3 \times 2^{k-1} \leq m < 3 \times 2^k$. If m happens to be a power of 2, say $2^{m'}$, then $2^{m'}$ is partitioned into $2^{m'-1}$ and $2^{m'-1}$ since in this case we have $k = m' - 1$. In fact, we have

$$\begin{aligned} 3 \times 2^{m'-2} &\leq 4 \times 2^{m'-2} = 2^{m'} \\ &= 2 \times 2^{m'-1} < 3 \times 2^{m'-1}. \end{aligned}$$

Thus, an interval of length 2^k is exactly halved in a way that $2^k \to 2^{k-1} \to \ldots \to 2 \to 1$.
On the other hand, if we have any other integer, then it is partitioned into a power of 2 and the rest in a manner described above. Because of the definition of the partition, the uniformity is at least 1/2. In fact, if $r = m - 2^k$ is greater than 2^k, then the uniformity is given by

$$2^k/(m - 2^k) \geq 2^k/(3 \times 2^k - 2^k) = 1/2,$$

and if r is at most 2^k then it is given by

$$(m - 2^k)/2^k \geq (3 \times 2^{k-1} - 2^k)/2^k = 1/2.$$

Thus, dividing the interval of length r is safe in the sense that it keeps the uniformity $\geq 1/2$. Dividing the interval of length 2^k is also safe since it is divided evenly. □

4 Known results

Some results are known for the problem defined on real numbers in a unit interval $[0, 1]$ instead of integers. In one dimension, an exact bound on the uniformity is known [1].

Theorem 2 *For any integer* $n > 0$ *there is a sequence of* n *points (real numbers) in a unit interval* $[0, 1]$ *with uniformity* $(\frac{1}{2})^{\lfloor n/2 \rfloor/(\lfloor n/2 \rfloor+1)}$ *and also any sequence of* n *points in the interval has uniformity at most* $(\frac{1}{2})^{\lfloor n/2 \rfloor/(\lfloor n/2 \rfloor+1)}$. *Such an optimal sequence can be computed in* $O(n)$ *time.*

Another important remark is that we can construct an optimal sequence if n, the number of points, is known in advance, but it is impossible otherwise.

Theorem 3 *For any integer* $n > 0$, *the greedy algorithm (Voronoi insertion) has uniformity* 1/2 *in a unit interval and* $\sqrt{2}/2$ *in a unit square.*

5 Uniform point sequence

Lemma 1 guarantees that for any pair of integers m and n there is a sequence of n integer points in the interval of length m such that its uniformity is at least $1/2$ if $m > n > 0$. What happens if we want to achieve uniformity strictly greater than $1/2$? First of all, we cannot expect the same property as in Lemma 1 any more. Suppose we are given an interval $[0, 9]$. Can we find a uniform sequence of 2 points achieving the uniformity strictly greater than $1/2$? The first division is uniquely determined as $\{9\} \to \{4, 5\}$ since $\{9\} \to \{3, 6\}$ has uniformity $3/6 = 1/2$. Now, we have to divide 5 into $\{2, 3\}$ since $\{1, 4\}$ is worse. Then, after the second division we have $\{4, 2, 3\}$ whose uniformity is exactly $1/2$. This simple example shows the difficulty of this extension. Now we have the following three problems.

Problem 1: Given two integers m and n with $m > n > 0$, determine whether there exists a strictly uniform (n, m)-sequence.

Problem 2: Given an integer $n > 0$, find the smallest integer m such that there is a strictly uniform (n, m)-sequence.

Problem 3: Given an integer $m > 1$, find the largest integer n such that there is a strictly uniform (n, m)-sequence.

In the problems above, a **strictly uniform** sequence means a sequence of points with uniformity strictly greater than $1/2$.

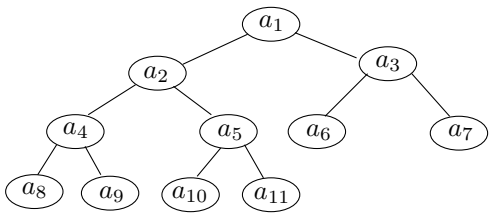
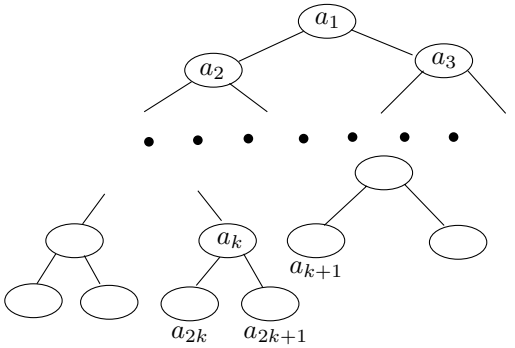
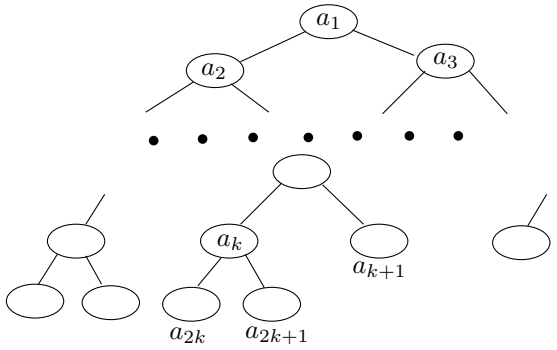


Figure 3: A tree describing the behavior of the algorithm in the proof. A node for an interval of length a_k is divided into two nodes for a_{2k} and a_{2k+1} with $a_k = a_{2k} + a_{2k+1}$. If a_k is the last node having children, then the first leaf node is a_{k+1} and the last one a_{2k+1}.

Let us first consider **Problem 2**. In this problem we look for a sequence $(p_1, \ldots, p_n)$ in an interval $[0, m]$ such that its uniformity is strictly greater than $1/2$. When we insert points $p_1, \ldots, p_n$ in order into the interval, then we can characterize the behavior of an algorithm by how the set of interval lengths changes. We start with the set $\{a_1\}$, where $a_1 = m$. Then, it is partitioned into a_2 and a_3 (we assume $a_2 \geq a_3$), and then a_2 is partitioned into a_4 and a_5 with $a_4 \geq a_5$.



(a) k is odd.



(b) n is even.

Figure 4: Two trees for odd and even integers. The node for a_{k+1} is a left or right child of its parent node depending on whether k is odd or even, respectively.

The first important observation here is that we have to partition the longest interval to keep the uniformity $> 1/2$. For dividing an interval that is not longest into two generates an interval of length shorter than half of the longest one, which results in uniformity $< 1/2$. If we always partition the longest interval, then the set of interval lengths is $\{a_{k+1}, \ldots, a_{2k}, a_{2k+1}\}$ after the k-th partition. Therefore, it is well described by a tree like a heap (see Figure 3).

In the algorithm we divide the intervals of lengths $a_1, a_2, \ldots$ in this order. When we divide a_k, a set of interval lengths is $\{a_k, a_{k+1}, \ldots, a_{2k-2}, a_{2k-1}\}$. Dividing a_k produces two new interval lengths a_{2k} and a_{2k+1}. Since we assume $a_{2k+1} \leq a_{2k}$ in our convention, a_{k+1} and a_{2k+1} are the maximum and minimum gaps after the division. Thus, we have

$$a_k = a_{2k} + a_{2k+1}, \text{ and}$$
$$a_{2k+1}/a_{k+1} > \tfrac{1}{2}, \quad \text{for} \quad k = 1, 2, \ldots, n.$$

When we are about to divide a_k, those interval lengths $a_k, a_{k+1}, \ldots, a_{2k-2}, a_{2k-1}$ are leaves of the corresponding tree. They are ordered in a way that $a_k \geq a_{k+1} \geq \cdots \geq a_{2k-2} \geq a_{2k-1}$.
Because of the uniformity condition $a_{2k-1}/a_k > 1/2$ must hold. Since the sum of those values $a_k + a_{k+1} +$

$\cdots + a_{2k-2} + a_{2k-1}$ is equal to a_1, the length of the original interval, we must minimize the sum to minimize the length of the original interval. What is the smallest value of a_{2k-1}? It depends on whether k is odd or even. See Figure 4.

Case 1: k is odd.

When a_{k+1} is to be divided, the set of interval lengths is $\{a_{k+1}, \ldots, a_{2k}, a_{2k+1}\}$. If we go back to the past divisions, $a_k, a_{k-1}, \ldots$ have been divided. When a_k was divided, we must have had $a_{2k-1}/a_k > 1/2$, that is, $a_{2k-1} > a_k/2$. Since $a_k = a_{2k} + a_{2k+1}$ and we assumed $a_{2k} \geq a_{2k+1}$, it means $a_{2k-1} > a_{2k+1}$. Therefore, a_{2k+1} may be equal to a_{2k}, but it must be strictly smaller than a_{2k-1}. Repeating this argument, we observe that the sum $a_{k+1} + \cdots + a_{2k+1}$ is minimized when $a_{2k+1} = a_{2k}$, $a_{2k} + 1 = a_{2k-1} = a_{2k-2}, \ldots, a_{k+3} + 1 = a_{k+2} = a_{k+1}$. Here note that the node of a_{k+1} is a right child since k is odd. Taking the constraint $a_{2k+1}/a_{k+1} > 1/2$ into accounts, we can conclude that $a_{2k+1} > (k-1)/2$, that is, a_{2k+1} must be at least $(k+1)/2$.

For the pattern we have

$$\begin{aligned} a_1 &= a_{k+1} + \cdots + a_{2k} + a_{2k+1} \\ &= (3k^2 + 4k + 1)/4. \end{aligned}$$

Case 2: k is even.

The proof proceeds similarly as above, but this time a_{k+1} is not paired. Considering the fact, we have

$$\begin{aligned} a_1 &= a_{k+1} + \cdots + a_{2k} + a_{2k+1} \\ &= (3k^2 + 6k + 4)/4. \end{aligned}$$

The results are summarized in the following theorem.

Theorem 4 *The length of the shortest interval that accepts of a uniform point sequence of length n is $(3n^2 + 4n + 1)/4$ if n is odd and $(3n^2 + 6n + 4)/4$ otherwise.*

Table 1. The longest uniform sequence for each length of an interval m.

m	2	3	4	5	6	7	8	9	10
n	1	-	1	1	1	2	2	1	3
m	11	12	13	14	15	16	17	18	19
n	3	2	2	3	3	3	3	3	4
m	20	21	22	23	24	25	26	27	28
n	4	4	3	3	5	5	5	4	4

In a similar manner we can characterize uniform sequences. Using the characterization, it is not so hard to solve the remaining problems. Due to space limit, we omit the details.

Table 1 shows the largest integer n, the length of the longest uniform sequence for each value of m, the length of the initial interval. There is no uniform sequence of length at least 1 for $m = 3$, which is indicated by the symbol $-$ in the table.

Table 2 shows the shortest interval that accepts a uniform point sequence of length n for each n.

Table 2. The shortest interval length accepting a uniform point sequence of length n.

n	1	2	3	4	5	6	7	8
m	2	7	10	19	24	37	44	61
n	9	10	11	12	13	14	15	16
m	70	91	102	127	140	169	184	217

6 Conclusions and future works

In this paper we have presented algorithms for generating uniform sequences of points in a given interval. One big difference from the existing study is that points must have integer coordinates. Due to this integrality the problem is now a combinatorial optimization problem. One important extension of our result is to higher dimensions, especially points sets in the plane. Although the problem has a complete solution in one dimension, no optimal solution has been known for point sets in the plane or space. The discrete version of the problem is expected to provide a combinatorial approach to the two-dimensional online discrepancy problem.

Acknowledgments

This work was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research on Priority Areas and Scientific Research (B).

References

[1] S. Teramoto, T. Asano, N. Katoh, and B. Doerr: "Inserting Points Uniformly at Every Instance," IEICE Trans. INF. & SYST., Vol. E89-D, 8, pp.2348-2356, 2006.

[2] T. Asano and S. Teramoto: "On-line uniformity of points," Book of Abstracts for 8th Hellenic-European Conference on Computer Mathematics and its Applications, pp. 21-22, Athens, Greece, September, 2007.

[3] B. Chazelle: "The Discrepancy Method: Randomness and Complexity," *Cambridge University Press,* 2000.

[4] J. Matoušek: "Geometric Discrepancy," *Springer,* 1991.

Edge-Unfolding Medial Axis Polyhedra

Joseph O'Rourke*

Abstract

It is shown that a convex medial axis polyhedron has two distinct edge unfoldings: cuttings along edges that unfold the surface to a simple planar polygon. One of these unfoldings is a generalization of the point source unfolding, and is easily established to avoid overlap. The other is a novel unfolding that requires a more complex argument to establish nonoverlap, and might generalize.

1 Introduction

Medial axis polyhedron. Let P be a convex polygon. The *medial axis* $M = M(P)$, $M \subset P$ is the closure of the locus of the centers of disks in P, each of whose boundary touches ∂P in two or more points. The medial axis is a well-studied construct that applies much beyond convex polygons, but we restrict our attention here to convex P. Then, M is a tree of straight segments whose leaves are the vertices of P. To each point $m \in M$ may be associated the radius $r(m)$ of the maximal disk in P centered on m. Let P lie in the xy-plane, and for each $m \in M$, define a point $p(m) = (m_x, m_y, r(m))$: it is vertically above m at height $z = r(m)$. Finally, define the *medial axis polyhedron* $\mathcal{P}$ for P to be the convex hull of $P \cup \{p(m) : m \in M\}$. See Fig. 1 for an example that we will use throughout. Let $\mathcal{M}$ be the tree of edges of $\mathcal{P}$ that project to M.

The medial axis polyhedron is studied in [5, p. 376]. An alternative construction is to define a halfspace through each edge of P that makes an angle of $\pi/4$ with respect to the xy-plane containing P, and includes P. The intersection of these halfspaces with $z \geq 0$ yields $\mathcal{P}$. One property established in [5] (for arbitrary piecewise-C^2 closed curves, not just convex polygons) is that the surface over the base is developable, i.e., it can be "developed" without distortion flat to a plane. However, in general developable surfaces develop with overlap. Here we are explicitly seeking a nonoverlapping development via cuttings along edges.

Source unfolding. The medial axis $M(P)$ is also known as the *cut locus* of ∂P: the closure of the locus of points with more than one distinct shortest path from ∂P. The points $\mathcal{M}$ on $\mathcal{P}$ have the same property, and so form the cut locus of the base rim ∂P measuring shortest paths on the surface. It is well known that the cutting the cut locus of a "source" point x on a convex polyhedron unfolds the surface to a nonoverlapping unfolding, the *source unfolding* [1, p. 359]. Cutting $\mathcal{M}$ on a medial axis polyhedron $\mathcal{P}$ is cutting the cut locus of ∂P, and it is easy to see that this leads to a nonoverlapping unfolding for medial axis polyhedra. For each face f_i incident to a base edge e_i can be viewed as composed of shortest paths to $\mathcal{M}$, each path a segment perpendicular to e_i. Cutting $\mathcal{M}$ permits each face to flip out, rotating about e_i into the xy-plane. The perpendicularity of the shortest segments to e_i and the convexity of P easily guarantee nonoverlap of this unfolding. This is also a special case of a "dome unfolding," which was already known to avoid overlap [1, p. 322].

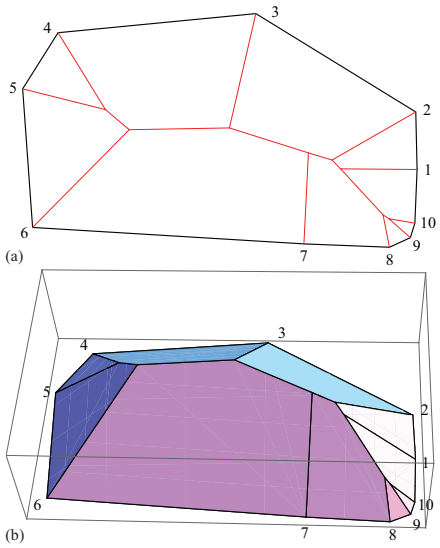


Figure 1: (a) A convex polygon P and its medial axis $M(P)$. (b) The corresponding medial axis polyhedron $\mathcal{P}$.

Convex cap unfolding. Of more interest is an unfolding that in some sense "squashes" the *convex cap* over P into the plane. Convex caps meet every line orthogonal to P in at most one point. They are an interesting special case to explore the long-unsolved problem of whether or not convex polyhedra always have an edge unfolding. One special case is studied

*Dept. Comput. Sci., Smith College, Northampton, MA 01063, USA. orourke@cs.smith.edu.

in [4]; the work here establishes another special case.

The research in [2] led to the conjecture[1] that cutting the cut locus of a simple, closed quasigeodesic leads to a nonoverlapping unfolding. As ∂P is such a quasigeodesic, unfolding two medial axis polyhedra glued base-to-base on the same P via the unfolding described in the next section establishes a (very) special case of this conjecture.

2 Unfolding

Unfolding defined. Let $(v_1, \ldots, v_n)$ be both the 2D vertices of P and the corresponding 3D vertices of $\mathcal{P}$; the context will disambiguate. Let $e_i = v_i v_{i+1}$ be the edges of P (and $\mathcal{P}$), let f_i be the face of $\mathcal{P}$ incident to e_i, and let u_i be the edge of $\mathcal{P}$ incident to v_i and shared between f_{i-1} and f_i. The unfolding U of $\mathcal{P}$ we study is obtained by cutting every edge of $\mathcal{M}$ not incident to a leaf vertex v_i, and cutting u_1, the edge of $\mathcal{M}$ incident to v_1. We ignore the base P for now; it is easily attached later. U consists of the faces $f_1, f_2, \ldots, f_n$ glued together at the shared edges u_i in a sequence. (See ahead to Fig. 2.) We view ∂U as composed of two parts: the *outer shell* constituted by the edges e_i of P, and the *inner path* constituted by images of cut edges of $\mathcal{M}$. We continue to call the vertices of the outer shell $v_1, \ldots, v_n$, with v_1' the second image of v_1.

Let α_i be the angle of P at v_i, and β_i the sum of the two (equal) angles of $\mathcal{P}$ incident to v_i in faces f_{i-1} and f_i. Thus β_i is the angle at v_i in U.

Lemma 1 *The outer shell of ∂U is a convex curve.*

Proof. *Sketch.* Calculation shows that

$$\beta_i = 2\cos^{-1}\left(\frac{\sqrt{2}\cos(\alpha_i/2)}{\sqrt{3-\cos\alpha_i}}\right)$$

and that $\alpha_i < \beta_i < \pi$. □

This ensures that P may be attached to U at any edge e_i and avoid overlap. Henceforth we concentrate on the nonoverlap of U.

Medial axis overlay. We close the outer shell of U into a convex region U^* by extending rays from v_2 through v_1, and from v_n through v_1'. If these rays do not meet, then U^* is unbounded. This indeed can occur (roughly, when α_1 is small), but the medial axis is easily defined for unbounded regions.

Define a *cell* of a medial axis $M(P)$ as one of the convex regions into which $M(P)$ partitions P, i.e., closures of the sets $P \setminus M(P)$. The key claim is the following:

[1] Made only in the presentation [3].

Theorem 2 *Each face f_i of U nests inside a cell of $M(U^*)$.*

We say f_i *nests* inside cell C_i if they share edge e_i and $f_i \subseteq C_i$. Because the cells of $M(U^*)$ partition U^*, this theorem implies nonoverlap of U. See Fig. 2.

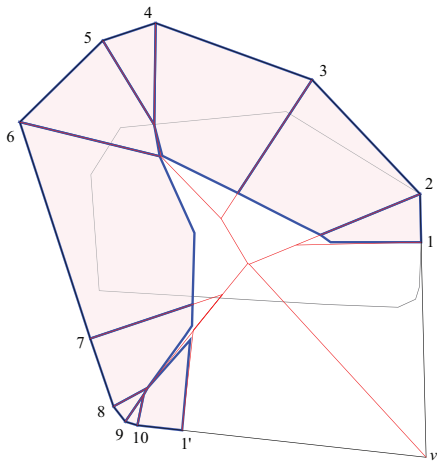


Figure 2: Unfolding U and polygon U^* for $\mathcal{P}$ in Fig. 1(b), overlaid with $M(U^*)$.

3 Inductive construction

Our proof of Theorem 2 relies on the well-known inductive construction of the medial axis for a convex polygon. $M(P) = M(P_n)$ is constructed by extending a pair of edges e_{i-1} and e_{i+1} to meet at $v_{i,j}$ and "engulf" e_i to create a superset polygon P_{n-1} of one fewer vertex, $(\ldots, v_{i-1}, v_{i,j}, v_{i+2}, \ldots)$. See Fig. 3. We

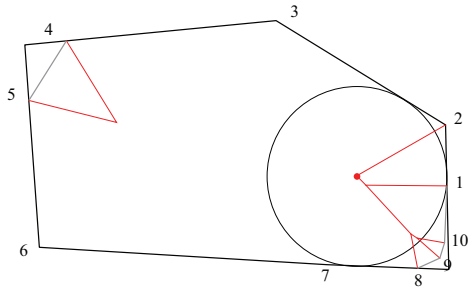


Figure 3: Partial inductive construction of $M(P)$ in Fig. 1(a).

study two unfoldings U_n and U_{n-1} that are based on polygons P_n and P_{n-1} related in just this manner. We will use primes or the subscript $n-1$ to distinguish the elements of U_{n-1} from the corresponding elements of U_n.

Lemma 3 *Let U_n and U_{n-1} be related by removing e_i from P_n, as described above. For $j \notin \{i-1, i, i+1\}$,*

the cell C'_j of $M(U^*_{n-1})$ nests inside the corresponding cell C_j of $M(U^*_n)$. For $j \in \{i-1, i+1\}$, the cells nest except for the portion cut away to remove e_i.

Here by "nests" we mean nests after a rigid movement that places e'_j and e_j into coincidence. With this lemma in hand, it will be straightforward to establish Theorem 2 by induction. We will use Fig. 4 to illustrate the proof. Here $e_i = e_4$ in U_{10} is removed to create U_9.

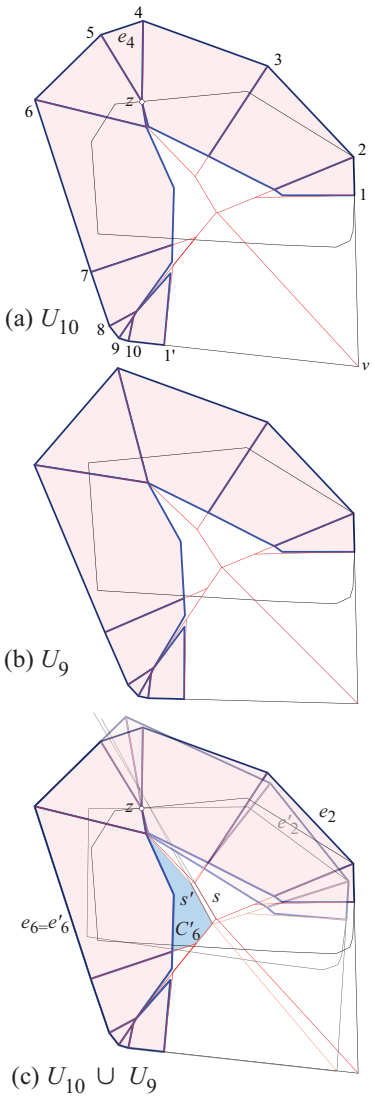


Figure 4: Edge $e_i = e_4$ is engulfed in the $U_{10} \to U_9$ transition. $e_j = e_6$ and $e_k = e_2$. C'_6 enlarges to C_6.

Proof. *Sketch.* If the boundary of the cell C'_j is composed of subsegments of bisectors of edges of U^*_{n-1} all indexed less than $i-1$ or all greater than $i+1$, then $C'_j = C_j$ and there is nothing to prove. In Fig. 4 this holds for $\{C_1, C_7, C_8, C_9, C_{10}\}$. So suppose C'_j's boundary contains a segment s' that is a bisector of e_j and e_k, where i lies between j and k. Let $v_{j,k}$ be the point of intersection of the extensions of these edges, through which the bisector containing s' passes. Let z be the vertex of U_n that is the apex of the triangle eliminated, $\triangle v_i v_{i+1} z$. See Fig. 4(a).

Claim 1. When U_{n-1} is positioned so that e'_j coincides with e_j, then z lies to the same side of a perpendicular line through s' as does $v_{j,k}$. See Fig. 4(c).

The segment s' of C'_j changes to s of C_j by a rotation of e'_k about z to e_k.

Claim 2. The rotation of the bisector of $b' = (e'_j, e'_k)$ containing s' to the bisector $b = (e_j, e_k)$ containing s, with $e'_j = e_j$ fixed, is such that s strictly expands C_j.

These two claims rely on technical lemmas described below. The consequence of Claim 2 (which relies on Claim 1) is that every segment of C'_j moves in such a way as to expand to C_j.

For $j \notin \{i-1, i, i+1\}$, this suffices to show that C'_j nests inside C_j. For $j \in \{i-1, i+1\}$, C'_j in fact does not nest in C_j, because C'_j includes $\triangle z v_i v_{i,j}$ or $\triangle z v_{i,j} v_{i+1}$, not present in C_j. Compare C_3 and C_5 in Figs. 4(a,b). However, $C'_j \setminus \triangle$ does nest in C_j (where $\triangle$ is the appropriate triangle), for the same reason: the segment s' rotates to s about z to enlarge the cell. □

3.1 Technical lemmas

Lemma 4 *Let s be a segment of the medial axis of a convex polygon P deriving from a maximal disk touching e_j and e_k, whose extensions meet at $v_{j,k}$. Then all points of the medial axis deriving from the portion of ∂P from e_j to $_k$ to the $v_{j,k}$-side is to that same side of any perpendicular line L through s.*

Proof. *Sketch.* See Fig. 5. □

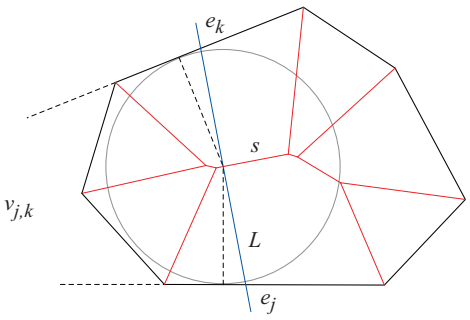


Figure 5: Lemma 4.

Lemma 5 *With $e'_j = e_j$ fixed, let b' and b be the bisectors of e_j with e'_k and e_k respectively, where e_k is a rotation of e'_k about a point z that lies between e_j and b'; see Fig. 6. Then the bisectors meet at a point $q = s' \cap s$ which is left of the line through z perpendicular to b'.*

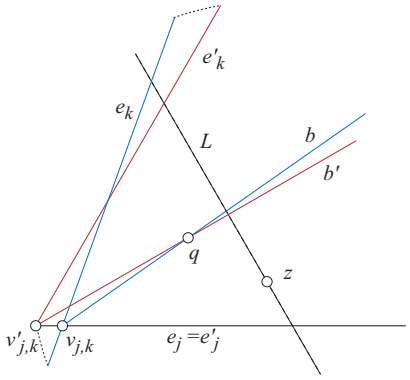


Figure 6: When e'_k rotates to e_k about z, $q = b' \cap b$ is left of L, the perpendicular to b' through z.

Proof. *Sketch.* Let e'_k rotate δ about z. If b' forms an angle of θ' with e_j, then after rotation of e_k, bisector b forms an angle $\theta > \theta'$ with e_j. One can show that $\theta = \theta' + \delta/2$.

Case 1. $v_{j,k}$ lies right of $v'_{j,k}$ as in Fig. 6. As z slides up a fixed L toward b', q moves up b' from $v'_{j,k}$, and $q = z$ at b'. Prior to that, q lies left of L.

Case 2. $v_{j,k}$ lies left of $v'_{j,k}$ on the line containing e_j. Then q falls behind $v'_{j,k}$ on b', well left of L. □

The reason that Lemmas 5 and 4 support the claims of Lemma 3 is as follows. Lemma 4 places z to the "correct" side of the endpoint of s'. Lemma 5 shows that the rotation about z that constitutes the $U_{n-1} \to U_n$ transition causes the bisectors $b \supset s$ and $b' \supset s'$ to meet at a point q even further to the z-side of the endpoint of s'. Thus, s is moved away from s' throughout its length, and so $C_j \supset C'_j$.

Completing the induction. Consider the construction sequence hinted at in Fig. 3: $P = P_n, P_{n-1}, \ldots, P_3$. Each polygon P_i leads to an unfolding U_i and medial axis $M(U_i^*)$. We know from Lemma 3 the cells of the $M(U_i^*)$ nest. So, starting from face f_j nested in C_j for some U_i, $i \geq 3$, the nesting will continue for all greater i, and thus establish the nesting claimed in Theorem 2. All that remains is establishing the base of this induction.

Lemma 6 *For P_3 a triangle, the three faces f_i of U_3 each nest inside the cell C_i of $M(U_3^*)$.*

Proof. *Sketch.* The apex z of $\mathcal{P}_3$ is equidistant from the three edges of P_3, and therefore z in U_3 is at the center of a circle that touches the three edges of U_3^*. See Fig. 7. □

4 Extensions

Pottmann and Walner consider in [5, p. 358ff] the more general polyhedron constructed by slanting

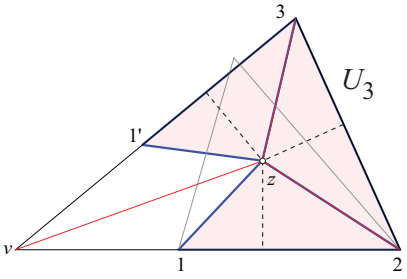


Figure 7: Induction base case: z is a vertex of the medial axis of U_3^*.

planes at some constant angle γ ("constant slope developable surfaces"). Call such a polyhedron $\mathcal{P}(\gamma)$; the medial axis polyhedron is $\mathcal{P}(\pi/4)$. It is not difficult to prove that the projection of $\mathcal{M}$ from $\mathcal{P}(\gamma)$ to the plane of P is independent of γ, i.e., it is always the medial axis $M(P)$. The following additional hypotheses appear to hold, although I have not yet proved them formally:

1. The main theorem (Theorem 2) holds for $\mathcal{P}(\gamma)$ for any γ and therefore shows all these polyhedra unfold without overlap in the same manner.
2. A polyhedron consisting of $\mathcal{P}(\gamma_1)$ and $\mathcal{P}(\gamma_2)$ glued base-to-base on the same P unfolds by gluing the convex outer shells of the two unfoldings along a common edge.
3. For any given γ, deform $\mathcal{P}(\gamma)$ by driving $\gamma \to 0$ continuously, meanwhile maintaining the original β_i face angles incident to each v_i, and allowing the faces to extend as needed to fill in the gaps at the "cut" edges. When $\gamma = 0$ is reached, the result is the unfolding U^*.

Finally, perhaps the analog of Theorem 2 holds for cutting the cut locus of an arbitrary convex cap, which would establish the quasigeodesic conjecture for convex caps.

References

[1] E. D. Demaine and J. O'Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge University Press, July 2007.

[2] J.-i. Itoh, J. O'Rourke, and C. Vîlcu. Unfolding convex polyhedra via quasigeodesics. Technical Report 085, Smith College, July 2007. arXiv:0707.4258v2 [cs.CG].

[3] J.-i. Itoh, J. O'Rourke, and C. Vîlcu. Unfolding convex polyhedra via quasigeodesics: Abstract. In *Proc. 17th Annu. Fall Workshop Comput. Comb. Geom.*, Nov. 2007.

[4] J. O'Rourke. Unfolding restricted convex caps. Technical Report 086, Smith College, September 2007. arXiv:0709.1647v1 [cs.CG].

[5] H. Pottmann and J. Wallner. *Computational Line Geometry*. Springer-Verlag, 2001.

Inducing Polygons of Line Arrangements

Elena Mumford* Ludmila Scharf† Marc Scherfenberg†

Abstract

We show that an arrangement $\mathcal{A}$ of n lines in general position in the plane has an inducing polygon of size $O(n^{4/3})$. Additionally we prove that $\mathcal{A}$ has a subarrangement of at least $2n/3$ lines with a linear size inducing polygon. We present an alternative algorithm for finding an inducing n-path for $\mathcal{A}$ and an algorithm that constructs an inducing n-gon for a special class of line arrangements.

1 Introduction

Every simple polygon induces an arrangement of lines, simply by extending its edges. We consider the question whether every arrangement of lines in the plane has an *inducing polygon*, namely, a simple polygon P such that every line l of the arrangement $\mathcal{A}$ is collinear with an edge of P and that every edge of P is collinear with some line of $\mathcal{A}$, see Fig. 6(right) for an example. There are arrangements that cannot be induced by a simple polygon. An arrangement of lines that all intersect in one point, and an arrangement of lines that form a 3×2 parallel grid serve as examples of such arrangements. However, we will show that when the lines of an arrangement are *in general position*, i.e., no three lines intersect in one point, and no two lines are parallel, the inducing polygon exists and can be found in $O(n^2)$ time. From now on we consider only arrangements of lines in general position in the plane.

A stronger version of the inducing polygon problem has been addressed by Bose et al. in [2]. Namely, the authors required the inducing polygon to be an n-gon, where n is the size of the arrangement. They showed that every arrangement of n lines contains an inducing simple n-path (also referred to as an *inducing polyline*), that is, a polygonal path that uses every line exactly once. Their algorithm produces an inducing n-gon if there exists a line such that all intersection points of the arrangement lie on one side of that line. Moreover, they demonstrated that every arrangement contains a subarrangement of size $\sqrt{n-1}+1$ with an inducing $(\sqrt{n-1}+1)$-gon.

In this paper we describe an $O(n^2)$ time algorithm for finding an inducing k-gon for an arrangement of n lines, where $k = O(n^{4/3})$. Additionally we present an alternative algorithm for finding an inducing n-path and an algorithm that constructs an inducing n-gon for any arrangement that maps to a convex set of points in dual space. Finally, we show that every arrangement of n lines has a subarrangement of size at least $2n/3$ with an inducing m-gon, where $m = O(n)$.

2 Algorithm based on the envelope polygon

In this section we describe an algorithm that constructs an inducing polygon P for every arrangement $\mathcal{A}$ of n lines.

Define the *envelope polygon* P_E of $\mathcal{A}$ as the polygon consisting of the finite length segments at the boundary of the unbounded faces of the arrangement [5]. A face of $\mathcal{A}$ is a *boundary face* if it is adjacent to an edge of P_E and is bounded, see Fig. 1(a) for an example, where the shading indicates the boundary faces of $\mathcal{A}$.

Our algorithm is based on the following observation: every line $l \in \mathcal{A}$ is either induced by P_E or crosses an edge of P_E. In the latter case l contains an edge of some boundary face f_l. Intuitively, the algorithm traverses the edges of P_E, scans every edge e for unused lines that cross e and for every such line l it augments the envelope polygon by "denting in" the face f_l as depicted in Fig. 1(b). More precisely, the

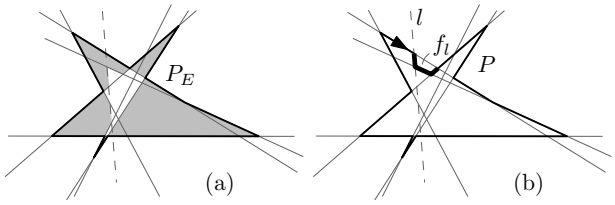


Figure 1: (a) The envelope polygon P_E with shaded boundary faces; (b) Augmenting the the polygon P for a line l.

algorithm constructs an inducing polygon P for $\mathcal{A}$ in the following way: First, set $P = P_E$. Then traverse the edges of P_E in clockwise order and scan every edge e for unused lines that cross it. The scanning direction for e is chosen according to the following rules:

(a) If both end vertices v_1 and v_2 of e are reflex, the scanning direction corresponds to the traversal direction of P_E, Fig. 2(a).

(b) If exactly one incident vertex of e is convex, the scanning direction is from the convex vertex to the

*Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, e.mumford@tue.nl

†Institute of Computer Science, Freie Universität Berlin, {scharf,scherfen}@mi.fu-berlin.de

reflex one, Fig. 2(b).
(c) If both end vertices are convex, find a line l' intersecting e, such that l' contributes a convex vertex x to P_E. We split e at the intersection point with l' and set the direction for two parts separately, each from the corresponding convex vertex towards the intersection point with l', Fig. 2(c).

Note that the line l' always exists due to the following observations: P_E has at least one convex vertex x that is not v_1 or v_2. From the definition of P_E and convexity of v_1 and v_2 it follows, that the intersection points of the line containing e with every other line of $\mathcal{A}$ lie between or at the vertices v_1 and v_2.

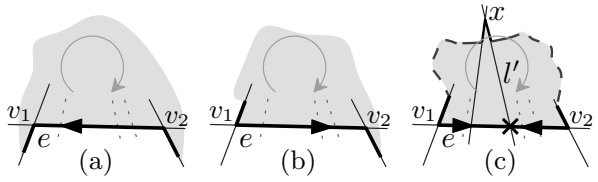


Figure 2: Direction rules for edge traversal. The shading indicates the interior side of P.

Next, for every unused line l that crosses e we consider the boundary face f_l that comes after l in the scanning direction of the edge. Let ∂f_l be the boundary of f_l and let e' be the edge of ∂f_l contained in e. We replace e' in P by $\partial f_l \setminus e'$ (see Fig. 1(b)) and mark all lines induced by $\partial f_l \setminus e'$, including the line l, as used.

Correctness. We need to show that P is a polygon, P induces $\mathcal{A}$, and that P is simple.

We start with $P = P_E$. Every augmentation replaces a line segment by a polygonal chain connecting its end points. Hence P is a polygon.

Every unused line crosses an edge of P_E. Therefore, all unused lines have been handled by the time the algorithm terminates. During each augmentation step only a part of an edge e of P_E is removed from P, so we never "lose" any of the lines that P induced before the augmentation. Thus, every line in $\mathcal{A}$ is induced by P.

We say that a polygon has a *self-intersection* if it has a pair of edges e_1 and e_2 intersecting in a single point v such that v is not an end point of either e_1 or e_2. A polygon has an *edge-overlap* if there exist two edges e_1 and e_2 in the polygon such that $e_1 \subseteq e_2$. A polygon has a *vertex-overlap* if it has two coinciding vertices, see Fig. 3 for examples. We can show that P does not have self-intersections, vertex- or edge-overlaps and hence is simple. Due to space limitations we omit the proof and refer the reader to the full version of the paper [6].

A trivial bound for the running time of the algorithm is $O(n^2)$. The bound for the complexity of the polygon is $O(n^{4/3})$ and is derived from the result by Clarkson et al. in [3].

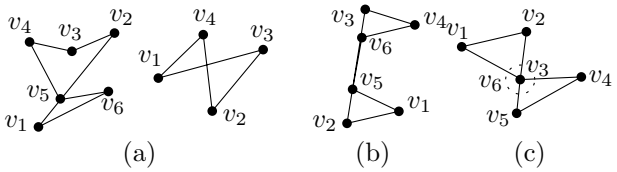


Figure 3: (a) Self-intersection; (b) edge-overlap; (c) vertex-overlap.

3 The Christmas tree algorithm

The Christmas tree algorithm constructs a simple inducing polyline for an arrangement $\mathcal{A}$ of size n that induces every line of $\mathcal{A}$ exactly once.

Given an arrangement of lines in the plane π our algorithm operates in the dual space π^* as defined in [4]: The dual of a point $p : (a, b) \in \pi$ is the line $p^* : f(x) = ax - b$ in π^*; the dual of a line $l : f(x) = ax + b$ in π is the point $l^* : (a, -b) \in \pi^*$. We can assume w.l.o.g. that $\mathcal{A}$ does not contain a vertical line.

Define L^* as a set of duals of lines of $\mathcal{A}$. Since the lines of $\mathcal{A}$ are in general position, L^* is in general position, i.e., no three points of L^* are collinear, and all points in L^* have different x-coordinates. Let M be an inducing polyline in π with n edges. We define a *dual polyline* for M as a polyline M^* in π^* that visits the vertices of L^* in the same order as M visits the lines of $\mathcal{A}$. M^* is uniquely defined by M; we omit the proof due to space limitations. Our algorithm finds a polyline P^* in π^* that visits every point of L^* exactly once such that the corresponding polyline P in π is simple.

The idea of the algorithm is to traverse all points in L^* and add them to P^* in a certain order. Let l_1^* be the bottommost point of L^* (if it is not unique, we take the rightmost of the two). We first construct a convex chain P_1^* by traversing the lower hull of L^* starting with l_1^* in descending order of the x-coordinate. The *lower hull* ([4]) of a set of points is defined as the part of the convex hull of the set that lies below or on the line connecting the leftmost and the rightmost points of the set. The *upper hull* is defined symmetrically. We add points of P_1^* to P^* except for the leftmost point of L^*, preserving the order of the vertices in P_1^*. Next, while we have unvisited vertices, we incrementally do the following.

- We construct a chain P_i^*, $i > 1$ by adding the vertices of the lower hull of $L^* \setminus P^*$ one by one in the order opposite to the order of vertices in P_{i-1}^*. That is, for odd i the vertices in P_i^* come in ascending order of their x-coordinate, and for even i vertices in P_i^* are in descending order of their x-coordinate.

- We add the vertices of P_i^*, except for the last one, to P^* preserving the order of vertices in P_i.

When we visited all the vertices in L^* we add the last point of the last chain to P^*. Fig. 4 depicts an example of the polyline P^* and the corresponding inducing polyline P in π. Note that the chains P_i^* resemble the garlands decorating a Christmas tree. A trivial upper bound for the running time of the algorithm is $O\left(n^2\right)$.

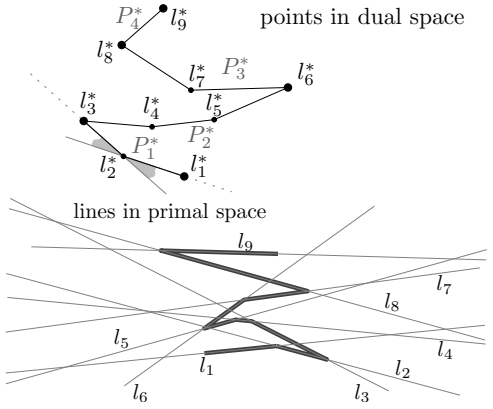


Figure 4: The path P^* generated by Christmas tree algorithm and the inducing path P for $\mathcal{A}$.

Correctness. By construction every point of L^* was added to P^* exactly once, hence P is an inducing n-path for $\mathcal{A}$. Next we demonstrate that P is simple.

Recall that a dual of segment s in π is a left-right double wedge in π^* bounded by the duals of the endpoints of s, see [4]. From now on we refer to such a double wedge as a *d-wedge* for short. Define a *center* of a d-wedge as the intersection point of the lines that bound it. We say that two d-wedges *intersect* if they contain a common line, see Fig. 5. In other words, two d-wedges intersect if and only if one contains the center of the other. It is known ([4]) that two segments in π intersect if and only if the corresponding d-wedges intersect in π^*.

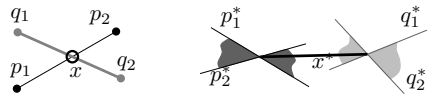


Figure 5: Intersecting segments and d-wedges.

Consider our polylines P and P^*. The edges of P correspond to d-wedges formed by pairs of consecutive edges of P^*. Thus the edges of P intersect if and only if the d-wedges formed by consecutive edges of P^* intersect. Hence from now on we are only interested in the d-wedges formed by consecutive edges of P^*. We call two such d-wedges *consecutive* if they are bounded by the same line. The d-wedges are consecutive if and only if they correspond to adjacent edges of P. Hence P is simple if and only if only consecutive d-wedges of P^* intersect.

Assume for a contradiction that we have two intersecting non-consecutive d-wedges w and v in P^*. Let p_1^* and p_2^* be the lines bounding w and let q_1^* and q_2^* be the lines bounding v. There are two cases to consider: (a) the centers of w and v are in the same chain P_i^* and (b) they belong to two different chains P_i^* and P_j^*, $i < j$.

Case (a): The center of v lies inside w, that is above p_1^* and below p_2^* (or the other way round). Since P_i^* is convex, this can only happen if the center of v lies on either p_1^* or p_2^*. Assume w.l.o.g. it lies on p_1^*. Since no three points in L^* are collinear, the center of v must be consecutive to the center of w along P^*. This means that w and v are consecutive, which contradicts our assumption.

Case (b): Recall that P_i^* is a lower hull of a set of points that includes the points of P_j^*. It means that for every edge e of P_i^* every point of P_j^* lies either above or on the line containing e. The center of v can lie inside of w if and only if it lies on either p_1^* or p_2^*. For the same reason as earlier it means that w and v are consecutive. We came to a contradiction again.

We can now conclude that P is indeed simple.

4 The Zigzag algorithm

When the duals L^* of the lines in $\mathcal{A}$ are in convex position, we can find an inducing n-gon of $\mathcal{A}$.

We first connect the leftmost point l^*_{left} and the rightmost point l^*_{right} in L^* with an edge e. The edge e divides the points into the upper hull H_u and the lower hull H_b of L^*. Starting with l^*_{left} we connect the the points of H_b in a zig-zag manner as illustrated in Fig. 6. Namely, connect l^*_{left} to a point $l_1^* \in H_b$ with the largest x coordinate. Then connect l_1^* to the next point l_2^* in $H_b \setminus \{l^*_{\text{left}}, l^*_{\text{right}}, l_1^*\}$ with the smallest x coordinate. We carry on connecting alternately the so far unvisited points with the largest and the smallest x-coordinate until all points of H_b are in the path. Next, starting with l^*_{right}, we traverse the upper hull in the "mirrored" way. Finally, we connect the end points of the obtained path with an edge and name the polygon P^*. The corresponding path P in primal space is an inducing n-gon for $\mathcal{A}$. The time complexity of the algorithm is $O(n \log n)$.

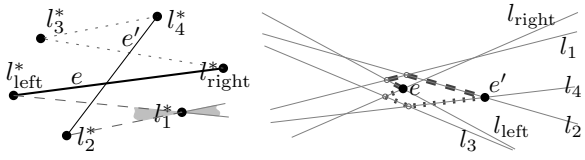


Figure 6: The polygons P^* and P produced by the Zigzag algorithm.

Correctness. During the traversal of the upper (lower) hull the algorithm visits alternately the so far unvisited points with the smallest and the highest x-coordinate. Hence, the x-coordinates of every three consecutively visited points p_1, p_2, p_3 form a bitonic sequence, that is the x coordinate of p_2 is either larger or smaller than the x-coordinates of both p_1 and p_2. Therefore, one part of the corresponding d-wedge is bounded by the segments $\overline{p_2p_1}$ and $\overline{p_2p_3}$ (*inner wedge*) and the other part by the extensions of these segments in the direction of p_2 (*outer wedge*). Obviously, the outer wedge cannot contain any other point because it is faced outside the convex hull. The inner wedge cannot contain any other point because p_1 and p_3 are neighbors on the convex hull. Hence, two non-consecutive wedges cannot share a common line. Since the two endpoints do not lie inside any other wedge, the closing edge also does not produce double-wedges causing intersections in primal space.

5 A linear inducing polygon for a subarrangement

Theorem 1 *An arrangement $\mathcal{A}$ of n lines has a subarrangement of size $s \geq 2n/3$ with an inducing polygon with at most $3.5n$ vertices or edges.*

Proof. Let n_E be the number of lines in $\mathcal{A}$ induced by the envelope polygon $P_E = E(\mathcal{A})$. We remove these n_E lines from the arrangement. We use the algorithm by Bose et al. to find an inducing polyline for the arrangement $\mathcal{A}'$ of n' remaining lines. Recall that the algorithm first constructs a polyline I' of size $n'-1$ that induces $n'-1$ lines of $\mathcal{A}'$ and has the following properties: (a) I' starts and ends on an unused line l of $\mathcal{A}'$; (b) the start and end points of I' are the extreme intersection points of I' and l on l, see Fig. 7(a). Next the algorithm revises I' if necessary and adds a segment of l to I' (see Fig. 7(b)) to obtain a polyline that induces $\mathcal{A}'$ and contains n' edges. Instead of this last step, we add the unbounded parts of l to the polyline I' (see Fig. 7(c)) to obtain the inducing polyline I with $n'+1$ edges, two of which are unbounded. This modification enables us to extend the first and the last segments of the inducing polyline to infinity while preserving its simplicity.

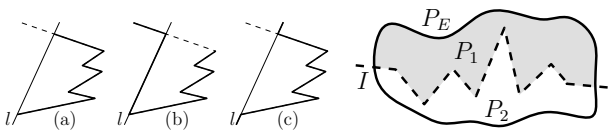


Figure 7: (a) The polyline I'; (b) inducing polyline by Bose et al.; (c) polyline I.

Figure 8: P_E cut by the polyline I into polygons P_1 and P_2.

We consider the polygon P_E and the polygons P_1 and P_2 formed by cutting P_E by the polyline I, see Fig. 8. P_E induces n_E lines, P_1 and P_2 induce n_1 and n_2 lines respectively. Let $P_{\max} \in \{P_E, P_1, P_2\}$ be the polygon that induces the largest subarrangement in $\mathcal{A}$. We can show that $n_1 + n_2 + n_E > 2n$, and $\max(n_E, n_1, n_2) \geq 2n/3$. Thus $P_{\max}$ induces at least $2n/3$ lines of $\mathcal{A}$. The size of $P_{\max}$ is bounded from above by n_E, $n_E < 3.5n$ ([1]). Thus $P_{\max}$ has at most $3.5n$ vertices or edges. $P_{\max}$ can be found in $O(n^2)$ time. □

6 Conclusions

We demonstrated that an arrangement $\mathcal{A}$ of n lines in general position in the plane has an inducing polygon of size $O(n^{4/3})$ that can be constructed in $O(n^2)$ time. Additionally we proved that $\mathcal{A}$ has a subarrangement of at least $2n/3$ lines with an inducing polygon of linear size. We also presented a new algorithm for finding a simple inducing polyline of size n for $\mathcal{A}$. Moreover, we showed that when the lines of $\mathcal{A}$ map to a set of points in convex position in dual space an inducing n-gon for $\mathcal{A}$ can be found in $O(n \log n)$ time. Thus we widened the class of arrangements for which an inducing n-gon can be constructed. The question whether such a polygon exists for an arbitrary arrangement of lines in general position remains open.

Acknowledgements. We thank Helmut Alt, Xavier Goaoc and Hyosil Kim for fruitful discussions.

References

[1] M. W. Bern, D. Eppstein, P. E. Plassman, and F. F. Yao. Horizon theorems for lines and polygons. In *Discrete and Computational Geometry: Papers from the DIMACS Special Year*, number 6 in DIMACS Ser. Discrete Math. and Theoretical Computer Science, pages 45–66. 1991.

[2] P. Bose, H. Everett, and S. Wismath. Properties of arrangement graphs. *Int. J. of Computational Geometry and Applications*, 13(6):447–462, 2003.

[3] K. L. Clarkson, H. Edelsbrunner, L. J. Guibas, M. Sharir, and E. Welzl. Combinatorial complexity bounds for arrangements of curves and spheres. *Discrete Comput. Geom.*, 5(2):99–160, 1990.

[4] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational geometry: algorithms and applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.

[5] D. Eu, E. Guevremont, and G. T. Toussaint. On envelopes of arrangements of lines. *Journal of Algorithms*, 21(1):111–148, 1996.

[6] L. Scharf. An inducing simple polygon of a line arrangement. Technical Report B 08-03, Freie Universität Berlin, 2008.

Coloring Geometric Range Spaces

Greg Aloupis* Jean Cardinal* Sébastien Collette†* Stefan Langerman‡* Shakhar Smorodinsky§

Abstract

Given a set of points in $\mathbb{R}^2$ or $\mathbb{R}^3$, we aim to color them so that every region of a certain family (for instance disks) containing at least a certain number of points contains points of many different colors. Using k colors, it is not always possible to ensure that every region containing k points contains all k colors. Thus, we introduce two relaxations: either we allow the number of colors to increase to $c(k)$, or we require that the number of points in each region increases to $p(k)$. We give upper bounds on $c(k)$ and $p(k)$ for half-spaces, disks, and pseudo-disks. We also consider the dual question, where we want to color regions instead of points. This is related to previous results of Pach, Tardos and Tóth on decompositions of coverings.

1 Introduction

We are interested in coloring finite sets of points in $\mathbb{R}^2$ or $\mathbb{R}^3$ so that any region (within a specified family) that contains at least some fixed number of points, also contains a significant number of distinctly colored points. For example, we study the following problem: *Does there exist a constant α such that given any set of points in the plane, it is always possible to color the points with k colors so that any halfplane containing at least αk points contains a point of each color?* In Section 2 we answer this question on the affirmative.

We also allow the number of available colors and the number of required distinct colors to be different. We ask, for instance, *Does there exist a constant α such that given a set of points in the plane, it is always possible to color the points with αk colors so that any halfplane containing at least k points also contains points of k distinct colors?* We show this is true as well. We ask similar questions for other types of regions such as disks and pseudo-disks

These types of problems can be seen as coloring range spaces induced by intersections of sets of points with geometric objects. The corresponding dual range spaces are those obtained by considering a finite set of regions in $\mathbb{R}^2$ or $\mathbb{R}^3$, and defining the ranges as the subsets of all regions containing a given point, for every possible point. The types of problems we ask when dealing with dual range spaces are analogous to the preceding questions. For instance: *Does there exist a constant α such that given any set of disks in the plane, it is always possible to color the disks with αk colors while ensuring that any point contained in at least k disks is contained in disks of k distinct colors?*

Definitions. A *range space* (or *hypergraph*) is a pair (S, R) where S is a set (called the *ground set*) and R is a set of subsets of S. Here, we consider finite restrictions of infinite geometric range spaces of the form $\mathcal{S} = (\mathbb{R}^d, \mathcal{R})$ for $d = 2$ or 3, where $\mathcal{R}$ is an infinite family of regions of $\mathbb{R}^d$. Such a finite restriction is a range space (S, R) where the ground set S is a finite set of points in $\mathbb{R}^d$ and the set of ranges R is the collection of subsets of S defined by the intersection of S with elements of $\mathcal{R}$: $R = \{S \cap r : r \in \mathcal{R}\}$.

We also consider the corresponding *dual range spaces*, denoted by $\widetilde{\mathcal{S}}$, of the form $\widetilde{\mathcal{S}} = (\mathcal{R}, \{r(p) : p \in \mathbb{R}^d\})$, where $r(p) = \{r \in \mathcal{R} : p \in r\}$ is the set of regions containing the point p. The finite restrictions of these dual range spaces are of the form $(S, \{r(p) \cap S : p \in \mathbb{R}^d\})$, where $S \subset \mathcal{R}$ is finite.

A *coloring* of a range space is an assignment of colors to the elements of the ground set. A c-coloring is a coloring that uses exactly c colors. A range is *k-colorful* if it contains at least k elements of distinct color. We are interested in the following two functions, for a range space $\mathcal{S}$:

1. $c_{\mathcal{S}}(k)$ is the minimum number for which there always exists a $c_{\mathcal{S}}(k)$-coloring of any finite restriction of $\mathcal{S}$, such that every range r is $\min\{|r|, k\}$-colorful.
2. $p_{\mathcal{S}}(k)$ is the minimum number for which there always exists a k-coloring of any finite restriction of $\mathcal{S}$ such that every range of size at least $p_{\mathcal{S}}(k)$ is k-colorful.

The goal of this paper is to provide upper bounds on $c_{\mathcal{S}}(k)$, $p_{\mathcal{S}}(k)$, $c_{\widetilde{\mathcal{S}}}(k)$, and $p_{\widetilde{\mathcal{S}}}(k)$ for various families of regions.

Previous results. The functions defined above are related to two previously studied problems. The first one is the decomposition of *f-fold coverings* in the plane: given a covering of the plane by a set of regions such that every point is covered by at least f regions, is it possible to decompose it into two disjoint coverings? This question was first asked by Pach in

*Université Libre de Bruxelles, CP212, Bvd. du Triomphe, 1050 Brussels, Belgium. Partially supported by the Communauté française de Belgique - ARC.{galoupis, jcardin, secollet, slanger}@ulb.ac.be

†Chargé de Recherches du FRS-FNRS.

‡Chercheur Qualifié du FRS-FNRS.

§Institute of Mathematics, Hebrew University, Givat-Ram, Jerusalem 91904, Israel. shakhar@cims.nyu.edu

1980 [4]. It is similar to deciding whether $p_{\widetilde{\mathcal{S}}}(2) \leq f$ for the dual range space $\widetilde{\mathcal{S}}$ defined by the considered family of regions, the difference being that we do not assume that all points are f-covered.

For $\mathcal{T}$ the range space defined by translates of a centrally symmetric convex polygon, Pach and Tóth [7] recently proved that $p_{\mathcal{T}}(k) = O(k^2)$ and $p_{\widetilde{\mathcal{T}}}(k) = O(k^2)$. Thus, a covering can be decomposed into k coverings if each point is covered at least ck^2 times for some constant c. On the negative side, for the range space induced by disks (denoted by $\mathcal{D}$), Pach *et al.* [6] proved that even $p_{\mathcal{D}}(2)$ is unbounded. They obtain a similar result for $p_{\widetilde{\mathcal{A}}}(2)$ where $\mathcal{A}$ is the family of either strips or axis-aligned rectangles.

The previous impossibilities constitute our main motivation for introducing some slack and defining the problem of $c(k)$-coloring a finite range space such that ranges are k-colorful, with $k \leq c(k)$.

The second previously studied problem is that of computing the chromatic number of geometric hypergraphs, defined as the minimum number of colors needed to make all ranges polychromatic, that is, 2-colorful [8]. It was shown that for the family of pseudo-disks $\mathcal{P}$, $c_{\widetilde{\mathcal{P}}}(2) = O(1)$.

A recent result of Chen *et al.* ([2], Thm. 3) also implies that $c_{\mathcal{A}}(k)$ and $p_{\mathcal{A}}(k)$ are unbounded, where $\mathcal{A}$ is the range space induced on $\mathbb{R}^2$ by axis-aligned rectangles. Furthermore, Pach and Tardos [5] proved a result implying $c_{\widetilde{\mathcal{A}}}(2) = \infty$, implying $c_{\widetilde{\mathcal{A}}}(k) = \infty$.

Our results. In Section 2, we consider the range space $\mathcal{H} = (\mathbb{R}^2, \mathcal{R})$, where $\mathcal{R}$ is the set of all halfplanes. We prove that $c_{\mathcal{H}}(k) \leq 3k-2$, and $p_{\mathcal{H}}(k) \leq 4k-1$. In other words, we can ensure that a halfplane contains k points of different colors in two ways: either we k-color the point set but require that the halfplane contains at least $4k-1$ points, or we allow the point set to be $(3k-2)$-colored.

In Section 3, we consider the range space $\mathcal{L} = (\mathbb{R}^3, \mathcal{R})$, where $\mathcal{R}$ is the set of all *lower halfspaces*. We prove that $c_{\mathcal{L}}(k) = O(k)$; and that $c_{\widetilde{\mathcal{L}}}(k) = O(k)$.

We provide a number of results on range spaces defined by disks and pseudo-disks in Section 4. For the range space $\mathcal{D}$ defined by disks, we prove that $c_{\mathcal{D}}(k) = O(k)$ by mapping disks in $\mathbb{R}^2$ to lower halfspaces in $\mathbb{R}^3$ and using the result of Section 3. For a dual range space $\widetilde{\mathcal{P}}$ defined by pseudo-disks we prove that $c_{\widetilde{\mathcal{P}}}(k) = O(k)$. Since halfplanes are a special case of pseudo-disks, we directly have $c_{\widetilde{\mathcal{H}}}(k) = O(k)$. We also show that $c_{\mathcal{P}}(k) = O(k)$, with similar arguments.

By lifting a 2D point set to the unit paraboloid $z = x^2+y^2$ in 3D, every lower halfspace in 3D isolates a set of points which is contained in a disk in the original set of points, and thus $p_{\mathcal{L}}(k) \geq p_{\mathcal{D}}(k)$. We also prove that $p_{\widetilde{\mathcal{L}}}(k) = p_{\mathcal{L}}(k)$: coloring lower halfspaces is equivalent in the projective dual to coloring points with respect to lower halfspaces.

All the proofs are constructive, and polynomial-time algorithms can be derived from them. The proofs of several lemmas are omitted in this abstract. The results are summarized in the following table, where the symbol $\star$ indicates new results; and the symbol ∞ indicates a function unbounded in terms of k.

$\mathcal{S}$	$c_{\mathcal{S}}(k)$	$p_{\mathcal{S}}(k)$	$c_{\widetilde{\mathcal{S}}}(k)$	$p_{\widetilde{\mathcal{S}}}(k)$
halfplanes	$\leq 3k-2$ Thm. 3$\star$	$\leq 4k-1$ Thm. 5$\star$	$O(k)$ Thm. 14$\star$	$\leq 8k-3$ Cor. 6$\star$
lower halfspaces in $\mathbb{R}^3$	$O(k)$ Thm. 10$\star$	∞ From disks	$O(k)$ Cor.11$\star$	∞ From disks
transl. of a cent. sym. cvx. poly.	$O(k)$ Thm. 16$\star$	$O(k^2)$ [7]	$O(k)$ Thm. 14$\star$	$O(k^2)$ [7]
disks	$O(k)$ Cor. 12$\star$	∞ [6]	$\leq 24k+1$ Cor. 15$\star$	
pseudo -disks	$O(k)$ Thm. 16$\star$	∞ [6]	$O(k)$ Thm. 14$\star$	

2 Halfplanes

In this section we study the case where the family $\mathcal{R}$ is the set of all halfplanes in $\mathbb{R}^2$. We denote by $\mathcal{H} = (\mathbb{R}^2, \mathcal{R})$ the corresponding infinite range space.

It is not always possible[1] to color a set of points S with k colors such that every halfplane of size k (containing k points of S) is k-colorful, even for $k = 2$. This is our main motivation for allowing either the number of colors or the range size to be greater than k.

We first need to recall the notion of Tukey depth:

Definition 1 *Given a set S of points in $\mathbb{R}^d$, the* Tukey depth *of a point p (not necessarily in the set) is the maximum integer t with the property that every halfspace containing p contains at least t points of S.*

It is well known that for any set of n points in the plane, there exists a point in $\mathbb{R}^2$ at depth $t \geq n/3$. The *depth-k region* is the set of all points at depth k or more. It is easily seen that this region is the intersection of all halfplanes containing more than $n-k$ points of S and thus its boundary is a convex polygon. We observe the following for depth-k regions:

Lemma 1 *Let S be a finite set of more than $3k$ points in $\mathbb{R}^2$. Then every open halfplane not intersecting the depth-k region of S and the bounding line of which is tangent to the depth-k region of S contains at most $2k-2$ points of S. The corresponding closed halfplane contains at least k points.*

We define the *orientation of a halfplane* as the absolute angle of the inward normal of the line bounding it. Thus, for example, the orientation of the halfplane defined by all points lying above the x-axis is $\frac{\pi}{2}$.

[1] The simplest example consists of an odd number of points in convex position.

Let p be a point of S lying outside the depth-k region. It is easily seen that the set of orientations of all closed halfplanes that are tangent to the depth-k region and that contain p form a closed (circular) interval of length at most π. Thus, each point may be represented as an arc on the unit circle. Let $\mathcal{A}$ be the set of arcs corresponding to points in S outside or on the boundary of the depth-k region, and let $\mathcal{A}'$ be the same set of arcs but open (in particular, degenerate arcs consisting of only one point are removed).

Lemma 2 *Every point on the unit circle is covered by at most $2k-1$ arcs of $\mathcal{A}'$, and every point that is not the endpoint of an arc is covered by at least k arcs. Furthermore, the minimum number of segments covering any point is at most $k-1$.*

Theorem 3 *$c_{\mathcal{H}}(k) \leq 3k-2$. That is, we can color any set of points in the plane with $3k-2$ colors such that any halfplane containing h points is $\min\{h,k\}$-colorful.*

Proof. A *proper coloring* of a set of arcs on the unit circle is an assignment of colors to the arcs such that no pair of arcs of the same color overlap. In [10] it was proved that every set of arcs on the unit circle has a proper coloring with $m+M$ colors, where m (resp. M) is the minimum (resp. maximum) number of arcs covering each point of the circle. Combining this with Lem. 2, we conclude that the corresponding set $\mathcal{A}'$ can be $(3k-2)$-colored. Accordingly we can color the points (outside the depth-k region) of S that correspond to $\mathcal{A}'$. The remaining points are colored arbitrarily. Thus there exists a $(3k-2)$-coloring of S such that every open halfplane not intersecting – but tangent to – the depth-k region is colorful (the colors of points inside that halfplane are pairwise distinct).

Now it remains to prove that every halfplane of size h is $\min\{h,k\}$-colorful. Given such a halfplane Π, there are two cases: (i) Π does not intersect the depth-k region, meaning that it is strictly contained in an open halfplane Π' which has its boundary line tangent to the depth-k region, and thus no two points in it are colored with the same color. (ii) Π intersects the depth-k-region and thus contains a closed halfplane Π' tangent to it. If the point p on the circle corresponding to Π' is not the endpoint of an arc, then Π' contains at least k points of different colors. If p is the endpoint of an arc then Π' contains at least all points corresponding to arcs that cover a point infinitesimally to the left of p, which also have at least k different colors. □

We now consider the depth-$2k$ region. As described above, points outside the depth-$2k$ region are associated with a set of closed arcs, $\mathcal{A}$, on the unit circle. Recall that each arc in $\mathcal{A}$ has length at most π and that by Lem. 1 every point on the unit circle is covered by at least $2k$ arcs.

Lemma 4 *Let $\mathcal{A}$ be a set of arcs of length at most π on the unit circle. If each point on the circle is covered at least $2k$ times then we can k-color the arcs of $\mathcal{A}$ so that each point on the circle is covered by k colors.*

Theorem 5 *$p_{\mathcal{H}}(k) \leq 4k-1$. That is, we can color any set of points in the plane with k colors such that any halfplane containing at least $4k-1$ points is k-colorful.*

Proof. Let $\mathcal{A}$ be the set of arcs corresponding to the points that lie outside or on the boundary of the depth-$2k$ region. By Lem. 4, $\mathcal{A}$ can be made k-colorful, as it covers every point of the unit circle at least $2k$ times. This means that there exists a k-coloring of S such that every closed halfplane tangent to the depth-$2k$ region is k-colorful. As we consider large point sets in comparison to k, there always exists a depth-$2k$ region (specifically, as long as $n \geq 6k$).

Let Π be a halfplane containing at least $4k-1$ points. Π must intersect (or touch) the depth-$2k$ region, because every open halfplane tangent to the region contains at most $4k-2$ points, by Lem. 1. Thus Π contains a closed halfplane Π' with its boundary tangent to the depth-$2k$ region. By construction, Π' must be k-colorful and therefore so must Π. □

Using projective duality, we obtain the following:

Corollary 6 *$p_{\mathcal{H}}^{\sim}(k) \leq 8k-3$. That is, we can color any set of halfplanes with k colors such that any point in the plane covered by $8k-3$ halfplanes is contained in halfplanes of k different colors.*

3 Lower halfspaces in $\mathbb{R}^3$

We now deal with the case where $\mathcal{R}$ consists of all *lower halfspaces* in $\mathbb{R}^3$. The proof of the following lemma is similar to that of Lem. 1 in $\mathbb{R}^2$. We call $\mathcal{L} = (\mathbb{R}^3, \mathcal{R})$ the corresponding infinite range space and consider the value of $c_{\mathcal{L}}(k)$. The depth-k region in $\mathbb{R}^3$ is bounded by a convex polyhedron.

Lemma 7 *Given a set of more than $4k$ points in $\mathbb{R}^3$, every open halfspace not intersecting the depth-k polyhedron and which has a bounding plane tangent to the depth-k polyhedron contains at most $3k-3$ points. The corresponding closed halfspace contains at least k points.*

We consider lower halfspaces defined by planes tangent to the depth-k polyhedron. Each normal vector to one of these planes corresponds to precisely one lower halfspace and defines one point on the unit sphere. We map the points from the unit sphere onto

the xy plane so that every lower halfspace corresponds to a single point in $\mathbb{R}^2$. This representation is used in the remainder of the section.

Lemma 8 *Let R_x denote the set of points in $\mathbb{R}^2$ corresponding to lower halfspaces tangent to the depth-k polyhedron and containing $x \in S$. Let p and q be two points of S outside the depth-k polyhedron. Then, R_x is a connected subset of $\mathbb{R}^2$, and the boundaries of R_p and R_q intersect at most twice.*

The proof of the next theorem uses the following definition and lemma [3]. We use the standard notion of chromatic number $\chi(G)$ of a graph G, defined as the minimum number of colors needed to color the vertices so that no edge is monochromatic.

Definition 2 *A simple graph $G = (V, E)$ is called k-degenerate for some positive integer k, if every (vertex-induced) subgraph of G has a vertex of degree at most k.*

Lemma 9 *Let $G = (V, E)$ be a k-degenerate graph. Then $\chi(G) \leq k + 1$.*

Theorem 10 *$c_{\mathcal{L}}(k) = O(k)$. That is, we can color any set of points in $\mathbb{R}^3$ with $O(k)$ colors such that any lower halfspace containing h points is $\min\{h, k\}$-colorful.*

Proof. Let $\mathcal{A} = \{R_x | x \in S$, outside or on the surface of the depth-k polyhedron$\}$. By Lem. 8, we know that $\mathcal{A}$ is a set of pseudo-disks. Let $\mathcal{A}'$ be the corresponding open pseudo-disks. By Lem. 7, we also know that every point in the projection of the sphere on $\mathbb{R}^2$ belongs to at most $3k - 2$ regions of $\mathcal{A}'$. We consider the intersection graph of $\mathcal{A}'$. Chan [1] observed that this graph is $O(k)$-degenerate, hence by Lem. 9, $O(k)$-colorable. □

Using projective duality again, we get the following:

Corollary 11 *$c_{\widetilde{\mathcal{L}}}(k) = O(k)$. That is, we can color any set of lower halfspaces in $\mathbb{R}^3$ with $O(k)$ colors so that any point in the intersection of more than k of them is covered by k different colors.*

4 Disks and pseudo-disks

In this section we consider the case where the ranges in $\mathcal{R}$ are disks or pseudo-disks. We denote by $\mathcal{D} = (\mathbb{R}^2, \mathcal{R})$ the range space for disks, and by $\widetilde{\mathcal{D}}$ its dual, where the ground set is the set of disks and the ranges are the subsets of all disks having a common point. Similarly, we use the notations $\mathcal{P}$ and $\widetilde{\mathcal{P}}$ for the range spaces defined by pseudo-disks.

The proof given above for lower halfspaces in $\mathbb{R}^3$ can be used to prove that $c_{\mathcal{D}}(k) = O(k)$, by using a standard lifting of the plane onto a parabola in $\mathbb{R}^3$.

Corollary 12 *$c_{\mathcal{D}}(k) = O(k)$.*

We now give a bound for the value of $c_{\widetilde{\mathcal{P}}}(k)$, where $\widetilde{\mathcal{P}}$ is the dual range space defined by pseudo-disks. Similar to the proof of Thm. 10, we analyze the degeneracy of a graph induced by a finite set of regions.

Definition 3 *Let S be a finite family of simple closed Jordan regions in $\mathbb{R}^2$. We denote by $G_k(S)$ the graph on S where the edges are all pairs $r, s \in S$ such that there exists a point p that belongs to $r \cap s$ and at most k other regions of S.*

The next lemma can be proved using a probabilistic technique similar to the one used in the classical proof of the crossing lemma [9].

Lemma 13 *Let S be a family of pseudo-disks. Then $G_k(S)$ is $O(k)$-degenerate, and hence the chromatic number of $G_k(S)$ is at most $O(k)$.*

Theorem 14 *$c_{\widetilde{\mathcal{P}}}(k) = O(k)$*

Corollary 15 *For the special case of real disks, it can be shown that the graph $G_k(S)$ is $24k$-degenerate. Hence in the special case of real disks, we have $c_{\widetilde{\mathcal{D}}}(k) \leq 24k + 1$.*

For the version in the primal range space in which we color points rather than regions, we can also prove the following, using a similar technique.

Theorem 16 *$c_{\mathcal{P}}(k) = O(k)$*

References

[1] T. M. Chan. Low-dimensional linear programming with violations. *SIAM J. on Computing*, 34(4):879–893, 2005.

[2] X. Chen, J. Pach, M. Szegedy, and G. Tardos. Delaunay graphs of point sets in the plane with respect to axis-parallel rectangles. manuscript, 2006.

[3] D. R. Lick and A. T. White. k-degenerate graphs. *Canadian J. on Mathematics*, 12:1082–1096, 1970.

[4] J. Pach. Decomposition of multiple packing and covering. In *2. Kolloq. über Diskrete Geom.*, pages 169–178. Inst. Math. Univ. Salzburg, 1980.

[5] J. Pach and G. Tardos. Personal communication. 2006.

[6] J. Pach, G. Tardos, and G. Tóth. Indecomposable coverings. In *CJCDGCGT 2005*, Lecture Notes in Computer Science, pages 135–148, 2007.

[7] J. Pach and G. Tóth. Decomposition of multiple coverings into many parts. In *Proc. of SoCG07*, pages 133–137, 2007.

[8] S. Smorodinsky. On the chromatic number of some geometric hypergraphs. *SIAM J. on Discrete Mathematics*, to appear.

[9] S. Smorodinsky and M. Sharir. Selecting points that are heavily covered by pseudo-circles, spheres or rectangles. *Combinatorics, Probability and Computing*, 13(3):389–411, 2004.

[10] A. Tucker. Coloring a family of circular arcs. *SIAM J. of Applied Mathematics*, 229(3):493–502, 1975.

A Lower Bound for the Transformation of Compatible Perfect Matchings

Andreas Razen*

Abstract

For a planar set of n points we consider the graph whose vertices are the crossing-free perfect matchings of the point set, and two such perfect matchings are adjacent if their union is also crossing-free. It was recently shown by Aichholzer et al. [2] that the diameter of this graph is in $O(\log n)$, improving over the previously best known upper bound of $n-2$.

We show a lower bound of $\Omega(\log n/\log\log n)$ for the diameter of this transformation graph of perfect matchings which nearly matches the upper bound. So far only constant lower bounds were known.

1 Introduction

Given a set P of n points in the plane let $T_{\text{pm}}(P)$ denote the set of all crossing-free straight-line perfect matchings of P. A straight-line embedded graph is called *crossing-free* if every pair of its edges does not share any point other than common endpoints. Two crossing-free perfect matchings M_1 and M_2 of P are *compatible* if their union, i.e., the graph on P with edge set $M_1 \cup M_2$, is crossing-free.

We are interested in the transformation graph $\mathcal{T}_{\text{pm}}(P)$ defined on the vertex set $T_{\text{pm}}(P)$ and with edges between compatible perfect matchings. Houle et al. [4] showed that for any set of n points this graph is connected and has diameter at most $n-2$. Recently, Aichholzer et al. [2] improved this upper bound to $O(\log n)$. So far no example was known for which the diameter is not constant. We give a sublogarithmic but rather tight lower bound of $\Omega(\log n/\log\log n)$. We do this constructively by providing point sets of increasing size, and on each point set we specify two perfect matchings achieving the bound.

Transformation graphs for various configurations have been treated in the literature. Well known for instance are the flip graphs for triangulations and pseudo-triangulations, or the tree graphs defined on the crossing-free spanning trees of the underlying point set with some predefined rule of transformation. In the case of transforming compatible spanning trees the asymptotics of the currently best-known upper bound ($O(\log n)$ due to Aichholzer et al. [1]) and lower bound ($\Omega(\log n/\log\log n)$ due to Buchin et al. [3]) for the diameter are the same as for perfect matchings.

*Institute of Theoretical Computer Science, ETH Zurich, razen@inf.ethz.ch

Although our construction for the lower bound when transforming perfect matchings uses similar ideas as in [3] note the problems' immanent difference of connectivity: Spanning-trees are connected graphs whereas perfect matchings consist of $n/2$ components. It is this property which makes the construction presented here more complicated.

2 The lower bound

In this section we construct planar point sets on which we specify pairs of perfect matchings which need a large number of steps to transform into each other via compatible perfect matchings, i.e., their distance in the transformation graph is large.

We start by introducing the concept of a *prisoner* which is a point serving as a witness that two perfect matchings have at least a certain distance in $\mathcal{T}_{\text{pm}}(P)$. Based on this we present a recursive construction in order to obtain point sets P of increasing size for which the diameter of $\mathcal{T}_{\text{pm}}(P)$ grows as well. This diameter is lower bounded by $\Omega(\log n/\log\log n)$, where n is the size of the underlying point set. For the sake of a simple description we use point sets with more than two points on a line, i.e., the point sets are not in general position. However, they can easily be changed to do so by applying a small perturbation without losing any of the construction's relevant properties.

The key idea is to consider two perfect matchings with a large number of crossings, in particular we will use a first matching with near horizontal edges and a second matching with near vertical edges. As intuitive as this approach may seem, having many crossings is not enough as can easily be seen by taking $n \geq 8$ points equidistantly distributed on a circle. Then the perfect matchings, one consisting of horizontal edges only and the other one of vertical edges only, have a large number of crossings. However, the diameter of the transformation graph is 2, since a perfect matching containing only edges on the boundary of the convex hull is adjacent to every perfect matching.

In order to deal with this issue we will impose dependencies onto the (near) horizontal edges such that whatever transformation is made certain (near) horizontal edges remain in the obtained matching.

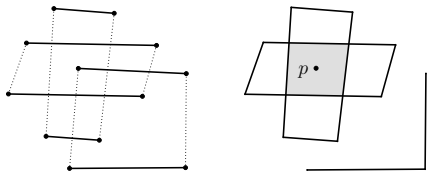
Before making this statement precise, we introduce some notation used in the following. The *granularity* of a point set P is the smallest positive difference of x-coordinates among points in P.

A *vertical strip* R is a subset of $\mathbb{R}^2$ such that there exist $a, b \in \mathbb{R}$ with

$$R = \left\{(x, y) \in \mathbb{R}^2 \middle| a \leq x \leq b\right\} =: [a, b] \times \mathbb{R};$$

the *width* of this strip is $b-a$. An edge *blocks a vertical strip* if the endpoints of the edge lie on different sides or possibly on the boundary of the strip.

Given two crossing-free perfect matchings M_1 and M_2 on a point set P, let $p \in P$ and consider the arrangement given by the set of edges in $M_1 \cup M_2$ not incident to p. Then p is called *prisoner w.r.t. M_1 and M_2* if there is a cell C of this arrangement such that $\overline{C} \cap P = \{p\}$, where $\overline{C}$ denotes the closure of C.



(a) M_1 and M_2 (b) Prisoner p

Figure 1: M_1 and M_2 have distance 3 in $\mathcal{T}_{\text{pm}}(P)$.

A prisoner guarantees a certain distance of the corresponding matchings in the transformation graph. Furthermore, as we will see later, given a fixed $d \in \mathbb{N}$ we can construct point sets such that after any d transformations the current matching and the target matching define a prisoner.

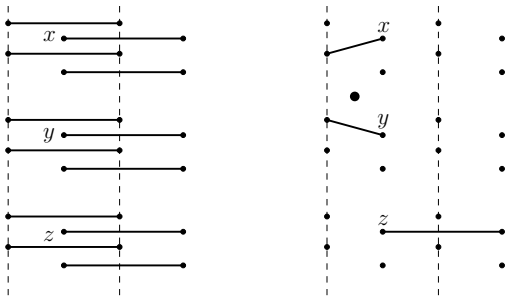
Lemma 1 *Let p be a prisoner w.r.t. M_1 and M_2. Then at least 3 steps are necessary to transform M_1 into M_2 by compatible perfect matchings.*

Proof. Observe that the existence of a cell C in the arrangement with $\overline{C} \cap P = \{p\}$ implies $M_1 \neq M_2$ and also that M_1 and M_2 are not compatible. Hence, their distance in $\mathcal{T}_{\text{pm}}(P)$ is at least 2. Assume it is exactly 2 then there is a perfect matching M compatible to both M_1 and M_2. However, this contradicts $\overline{C} \cap P = \{p\}$ since M matches p to some point outside $\overline{C}$. □

2.1 A first recursive construction

In the following we describe a way to construct point sets and two perfect matchings whose distance in the transformation graph can be made arbitrarily large. For this purpose consider the point set shown in Figure 2(a) given by three copies of a so-called *base gadget* together with a perfect matching consisting of horizontal edges only. The point set has granularity 1/2 assuming a proper coordinate system such that the vertical strip indicated by dashed lines is $[0, 1] \times \mathbb{R}$.

After 1 transformation step to a compatible perfect matching the edges leaving the points x, y and z block vertical strips of width at least 1/2, see Figure 2(b).

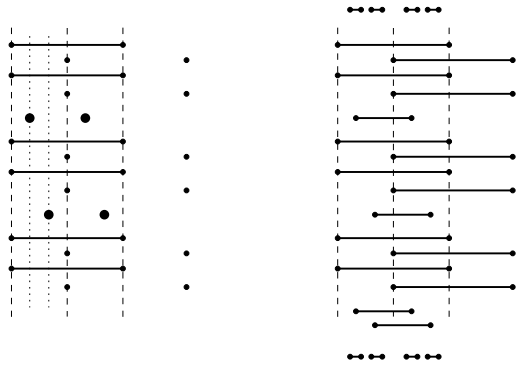


(a) Three copies of the base gadget (b) After the first transformation

Figure 2: Creating a prisoner.

Hence, by the pigeon-hole principle one of the vertical strips $[0, 1/2] \times \mathbb{R}$ or $[1/2, 1] \times \mathbb{R}$ is blocked twice. Now, placing a further point in-between the two blocking edges creates a candidate for a prisoner; we only need the vertical edges of a second matching in order to induce a corresponding cell. We define this second matching at the end of the discussion.

Since there are many ways to transform the initial perfect matching we have to make sure that for every possible pair of edges that block the same strip there is a candidate prisoner in-between. Therefore, we place points between the base gadgets that equidistantly subdivide each of the vertical strips of width 1/2 into three smaller strips, see Figure 3(a). It is crucial that these new points all have distinct x-coordinates since otherwise we might not obtain prisoners by adding the vertical edges of the second matching.

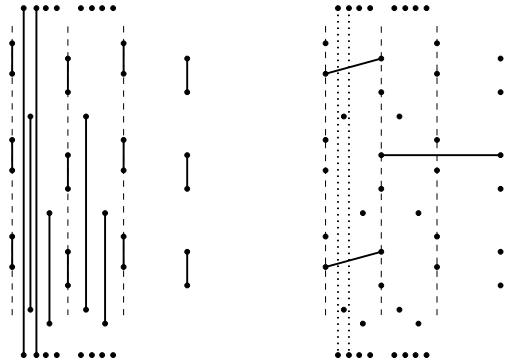


(a) Adding all candidate prisoners (b) The horizontal matching

Figure 3: The final point set.

In order to define the second perfect matching of vertical edges we introduce a matching partner for every candidate prisoner and place it below the so far constructed point set with the same x-coordinate. We call the hereby obtained set A which will be used in the recursive construction. Moreover, every candidate prisoner still lacks the two already mentioned vertical edges we need for a prisoner's cell.

The points for these edges are placed at the very top and the very bottom of A. Figure 3(b) shows the point set A together with the top- and bottom-most points and also the first horizontal matching. Figure 4(a) shows the second matching, where for the sake of readability only the first matching edges of top- and bottom-most points are drawn.



(a) The vertical matching

(b) A prisoner after 1 transformation

Figure 4: Point set achieving diameter 4.

Note that in any matching M compatible with the horizontal matching, there is a prisoner w.r.t. M and the vertical matching, see for instance Figure 4(b). Hence, by Lemma 1 the diameter of the corresponding transformation graph is at least 4.

In order to continue recursively with this idea we need to construct a point set with a prisoner after 2 transformation steps. For the sake of a readable presentation we refrain from specifying concrete coordinates for the constructed point sets and rather focus on explaining the construction more carefully.

Recall that we placed the candidate prisoners in such a way that only their matching partners have the same x-coordinate. Hence, a prisoner after 1 transformation cannot connect to its partner in the next step. By construction A has granularity $1/6$. Therefore, after 2 transformation steps there is a vertical strip of width at least $1/6$ which is blocked. This is because we placed the candidate prisoners equidistantly in-between the base gadgets.

Now, we vertically stack seven copies of A. By the pigeon-hole principle we know that for at least one $i = 1, \ldots, 6$ the strip $[(i-1)/6, i/6] \times \mathbb{R}$ is blocked twice after 2 transformations. Hence, we are left with defining the new candidate prisoners (and their corresponding matchings partners). We have to make sure that for every possible pair of blocking edges after 2 transformation steps there is a single candidate prisoner in-between the blocking edges. We achieve this in the following way: for $i = 1, \ldots, 6$ we separately consider the strip $[(i-1)/6, i/6] \times \mathbb{R}$ in which we place six points, one in-between each copy of A, equidistantly distributed inside $[(i-1)/6, i/6]$.

Hence, in total we add 36 candidate prisoners and equally many matching partners. Note that this new point set, call it B, has granularity $1/(6 \cdot 7) = 1/42$. Similarly to Figure 4(a), we add top- and bottom-most points for each candidate prisoner which we need for defining the prisoner's cell; recall again that these points are not part of the recursive construction.

Starting with the horizontal matching, after any 2 compatible transformation steps we obtain a new perfect matching which induces a prisoner together with the vertical perfect matching. Hence, the transformation graph has diameter at least 5.

By the same argument, stacking 43 copies of B and spreading in all candidate prisoners we obtain a new point set C which in turn, by adding top- and bottom-most points as before, yields a transformation graph with diameter at least 6. Note that the granularity of C has already decreased down to $1/(42 \cdot 43) = 1/1806$.

We omit the exact calculation of the diameter's asymptotic behavior in terms of the number of points used in this construction because we will drastically improve on it in the following section. However, note the doubly exponential decrease of the granularity and accordingly the doubly exponential growth of the number of previously constructed point sets used in the recursion. Then this construction leads to point sets of size n with perfect matchings M_1 and M_2 such that if d steps are needed to transform M_1 into M_2 then $n \in O\big(2^{2^d}\big)$, that is $d \in \Omega(\log \log n)$.

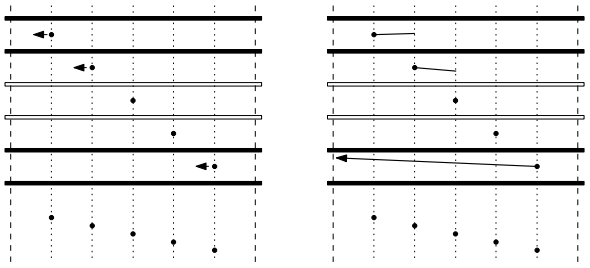
2.2 More prisoners help

In the following we will further develop the concept of the previous section for constructing point sets whose transformation graphs have large diameter. Recall that the sufficient condition for applying the pigeon-hole principle in the recursion is that the number of copies of previously constructed point sets is strictly larger than the inverse of the granularity.

We will now subdivide the vertical strips by even more candidate prisoners (still equidistantly) in order to reduce the number of copies we need to increase the diameter of the transformation graph by 1. This is motivated by the following. We did not yet take into account that (a lot) more than one strip may be blocked (a lot) more than just twice, which clearly happens when stacking more copies. Still every candidate prisoner in-between two blocking edges cannot connect to its partner after the next transformation step and hence will be incident to an edge blocking a strip of width at least the current granularity.

In order to make this statement precise we consider a single strip that is blocked at least twice and analyze what happens in the next transformation. For every pair of consecutive blocking edges we make the candidate prisoner in-between with the largest y-coordinate *responsible* for the small vertical strip to its left.

We show in the following that there is an injective map from the set of responsible candidate prisoners to the set of edges blocking the smaller strips after the next transformation step.



(a) Responsible points (b) Many blocked strips

Figure 5: What happens in one large strip?

In the example of Figure 5(a) we have a single strip marked by dashed vertical lines to the left and right. The six copies of the previous recursive construction step are shown as rectangles: they are empty if there is no edge blocking the strip, and full if the strip is blocked. The five prisoners between the copies subdivide the strip into six smaller strips indicated by dotted vertical lines. The arrows mark the prisoners responsible for their corresponding strip to their left.

After the next transformation step none of the responsible prisoners connects to its matching partner. Thus (considering the granularity of the point set) their incident edges either block the smaller strip to their left or to their right, see Figure 5(b). Note that the candidates are responsible for their *left* smaller strip; if they connect to the right then their left strip remains available unless a previous candidate already claimed it. By construction, once a responsible prisoner connects to the left its incident edge has to block *all* smaller strips until the very left end, and at least one of these small strips was not yet accounted for.

In particular this implies that every (except the topmost) edge which blocks the initial strip guarantees a blocked strip after the next transformation of width equal to the granularity. We are now ready to explain the more sophisticated recursive construction using this observation. We denote the base gadget from the previous section by S_0, and for $i \geq 1$ let $a_i \in \mathbb{N}$ be the number of copies of S_{i-1} we use to construct S_i. Denote $\pi_k = \prod_{i=1}^{k} a_i$, for $k \geq 1$. As before, from S_k one can easily construct a point set and define the corresponding horizontal and vertical matchings such that after k transformations there exists a prisoner.

In S_0 there are 2 strips, 1 of which is blocked after 1 step. Using the observation above, in S_1 there are $2a_1$ strips, of which at least $a_1 - 2$ are blocked after 2 steps. In S_2 there are $2a_1a_2$ strips, of which at least $(a_1-2)\cdot a_2 - 2a_1$ are blocked after 3 steps. Inductively, we find that in S_k there are $2\pi_k$ strips, of which at least $\pi_k - 2\sum_{i=1}^{k} \frac{\pi_k}{a_i}$ are blocked after $k+1$ steps. Note that in order for the construction to work we need to be able to apply the pigeon-hole principle in each recursion step. By induction, the last condition

$$\pi_k - 2\sum_{i=1}^{k} \frac{\pi_k}{a_i} = \pi_k\left(1 - 2\sum_{i=1}^{k} a_i^{-1}\right) > 0, \qquad (1)$$

is necessary and sufficient for this purpose. Now, we turn to the number of points used in the construction. Note that in addition to the copies of previously constructed point sets we also add candidate prisoners in-between. Let $n_i := |S_i|$ for $i \geq 0$, and recall that $n_0 = 8$. Then, clearly we have $n_1 = a_1 \cdot n_0 + 2(a_1 - 1)$ and $n_2 = a_2 \cdot n_1 + 2a_1(a_2 - 1)$. By induction we find

$$n_k = a_k n_{k-1} + 2\pi_{k-1}(a_k - 1) = (n_0 + 2k)\pi_k - 2\sum_{i=1}^{k} \frac{\pi_k}{a_i}.$$

With $a_i = 2d + 1$ for $i = 1, \ldots, d$, condition (1) holds. Hence, from S_d we obtain a prisoner after d transformations. Furthermore, there is a constant c with

$$n_d = (n_0 + 2d)(2d+1)^d - 2d(2d+1)^{d-1} = O\big(c^d d^d\big).$$

Adding top- and bottom-most points needed for the vertical edges of the prisoners' cells we obtain a point set with at most $2n_d$ points for which the diameter of the transformation graph is $d + 3$.

Theorem 2 *For arbitrarily large n there is a point set P of n points in the plane for which the diameter of $\mathcal{T}_{\mathrm{pm}}(P)$ is in $\Omega(\log n / \log\log n)$.*

Note that the choice of the a_i is optimal here, since (1) implies both $n_d > \pi_d$ and $1 > 2\sum_{i=1}^{d} a_i^{-1}$. The geometric-harmonic means inequality yields

$$\sqrt[d]{\pi_d} \geq \frac{d}{\sum_{i=1}^{d} a_i^{-1}} > 2d.$$

Acknowledgments

We would like to thank Dominik Scheder for reading a draft of the paper and for several helpful discussions.

References

[1] O. Aichholzer, F. Aurenhammer, C. Huemer, and H. Krasser, Transforming spanning trees and pseudotriangulations. *Inf. Process. Lett. 97*, 1 (2006), 19–22.

[2] O. Aichholzer, S. Bereg, A. Dumitrescu, A. García, C. Huemer, F. Hurtado, M. Kano, A. Márquez, S. Smorodinsky, D. Souvaine, J. Urrutia, and D. R. Wood, Compatible Geometric Matchings, (2007). *arXiv.org:0709.3375*

[3] K. Buchin, A. Razen, T. Uno, and U. Wagner, Transforming Spanning Trees: A Lower Bound, *Proc. 23rd Europ. Workshop on Comp. Geom.*, (2007), 166-169.

[4] M. E. Houle, F. Hurtado, M. Noy, and E. Rivera-Campo, Graphs of Triangulations and Perfect Matchings. *Graph. Comb. 21*, 3 (2005), 325–331.

Edge-Removal and Non-Crossing Configurations in Geometric Graphs

Oswin Aichholzer* Sergio Cabello† Ruy Fabila-Monroy‡ David Flores-Peñaloza‡ Thomas Hackl*
Clemens Huemer § Ferran Hurtado § David R. Wood §

Abstract

We study the following extremal problem for geometric graphs: How many arbitrary edges can be removed from a complete geometric graph with n vertices such that the remaining graph still contains a certain non-crossing subgraph. In particular we consider perfect matchings and subtrees of a given size. For both classes of geometric graphs we obtain tight bounds on the maximum number of removable edges. We further present several conjectures and bounds on the number of removable edges for other classes of non-crossing geometric graphs.

1 Introduction

A geometric graph is a graph $G = (V, E)$ drawn in the plane, such that V is a point set in general position (meaning that no three points of V lie on a common line) and E is a set of straight-line segments whose endpoints belong to V. A geometric graph is called *non-crossing* if no two edges intersect in their interior, but two edges might have an endpoint in common. Two edges are *disjoint* if they have no point in common.

Extremal problems for geometric graphs have received considerable attention. One problem considered in this area, studied by Erdős, Perles, Kupitz, and Avital and Hanani [1, 11], is to determine the smallest number $e_k(n)$ such that every geometric graph with n vertices and $m > e_k(n)$ edges contains $k + 1$ pairwise disjoint edges. Erdős [5] proved that $e_1(n) = n$. For three pairwise disjoint edges, bounds on $e_2(n)$ were given in [1, 7], culminating in $e_2(n) = 2.5n$ (plus a constant), as shown recently by Černý [3]. Bounds for $e_3(n)$ have been obtained in [7, 14].

For general values of k, Goddard et al. [7] showed that $e_k(n) \leq cn(\log n)^{k-4}$ for some constant c. This was improved by Pach and Törőcsik [12] to $e_k(n) \leq k^4 n$, the first upper bound linear in n. Tóth and Valtr [14] further improved this bound to $e_k(n) \leq k^3(n + 1)$, and finally Tóth [13] showed that $e_k(n) \leq 2^9 k^2 n$, where the constant 2^9 has since been improved by Felsner [6] to 256. Kupitz proved a lower bound of $e_k(n) > kn$, which was improved to $e_k(n) \geq \frac{3}{2}(k-1)n - 2k^2$ by Tóth and Valtr [14]. It is conjectured that $e_k(n) \leq ckn$ for some constant c.

Research on $e_k(n)$ has focused on small values of k. But k can be as large as $\frac{n}{2} - 1$, in which case we obtain a non-crossing perfect matching. Looking at the problem from this angle, we ask for $e_{n-1}(2n)$, or in other words, we investigate how many (arbitrary) edges can be removed from a complete geometric graph, such that it still contains a non-crossing perfect matching. We show that every complete geometric graph on $2n$ vertices still contains a non-crossing perfect matching after removing any set of $n - 1$ edges; that is $e_{n-1}(2n) = \binom{2n}{2} - n$. This bound is achieved for complete geometric graphs on point sets in convex position, meaning that there exists a set of n edges whose removal disallows a non-crossing perfect matching in the remaining graph. For point sets in convex position this question was completely settled by Kupitz and Perles for each k. They showed that if a geometric graph on n vertices in convex position has at least $(k-1)n + 1$ edges then the graph contains k disjoint edges, and this bound is tight; see [7].

Our research was motivated by a closely related problem posed by Micha Perles in 2002 and studied by Černý, Dvořák, Jelínek and Kára [4]: How many arbitrary edges can be removed from a complete geometric graph on n vertices such that the remaining graph still contains a non-crossing Hamiltonian path. It is of interest to study this problem for other classes of non-crossing geometric graphs. We consider subtrees of a given size. For the case of spanning trees, removing $n - 2$ arbitrary edges from any complete geometric graph on n vertices leaves a graph that still contains a non-crossing spanning tree [9]. Removal of more edges is possible if the set of removed edges has certain properties. Benediktovich [2] recently showed

*Institute for Software Technology, Graz University of Technology, Austria, {oaich,thackl}@ist.tugraz.at. Supported by the Austrian FWF Joint Research Project 'Industrial Geometry' S9205-N12.

†Department of Mathematics, IMFM, and Department of Mathematics, FMF, University of Ljubljana, Slovenia, sergio.cabello@fmf.uni-lj.si. Supported by the Slovenian Research Agency, project J1-7218.

‡Instituto de Matemáticas, Universidad Nacional Autónoma de México, ruy@ciencias.unam.mx, dflores@math.unam.mx.

§Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Barcelona, Spain, {clemens.huemer,ferran.hurtado,david.wood}@upc.edu.
Research supported by projects MEC MTM2006-01267 and Gen. Cat. 2005SGR00692. The research of David Wood is also supported by a Marie Curie Fellowship from the European Commission under contract MEIF-CT-2006-023865.

that each complete geometric graph on $n \geq 5$ vertices still contains a non-crossing spanning tree after removing any self-crossing 2-factor, i.e., a 2-regular spanning subgraph with two edges sharing an interior point. We show that every complete geometric graph on n vertices still contains a non-crossing subtree that spans $n-k$ vertices after removing $\lceil \frac{kn}{2} \rceil$ arbitrary edges, for $k \geq 2$, and this bound is tight.

Examples bounding the number of removable edges often are defined on point sets in convex position. We conjecture that for point sets with many points in the interior of the convex hull many more edges can be removed from the complete geometric graph to still guarantee the considered subgraph. We finally briefly consider this problem for other classes of geometric graphs.

In the following, removal of a set E' of edges of a complete geometric graph G is expressed by $G - H$, where E' is the edge set of a subgraph H of G. The edges of E' are called *removed* or *forbidden* edges. We omit several proofs in this abstract.

2 Perfect matchings

In this section we investigate the maximum number of removable edges in a complete geometric graph such that the remaining graph contains a non-crossing perfect matching. We first show a result for abstract graphs.

Theorem 1 *For all $p \geq 2$, for every spanning subgraph $H = (V, E')$ of the complete graph K_{kp} with $|E'| \leq k-1$, the graph $K_{kp} - H$ contains the complete p-partite graph $K_{k,\ldots,k}$.*

Proof. For each p we prove the theorem by induction on k. For $k = 1$ the statement is trivial. Assume the statement is true for every number $k' < k$. Now, we are given the complete graph K_{kp} and we are given a spanning subgraph $H = (V, E')$ with $|E'| \leq k-1$. Assume that $|E'| > 0$, as otherwise nothing has to be proved. Observe that there exists a set Q of at least $p-1$ isolated vertices in H and there exists a vertex $v \notin Q$ whose degree is at least 1 in H. Let $N(v)$ denote the set of neighbors of v in H. Define a graph $H' = (V \backslash (Q \cup \{v\}), E^*)$ where E^* is obtained by first taking the set of edges of the induced subgraph of $(V \backslash (Q \cup \{v\}), E')$ and then adding a minimum number of edges to the resulting set, such that $N(v)$ is connected. We have $|E^*| \leq |E'| - 1 \leq k-2$, because we removed $\deg_H(v)$ edges and added at most $\deg_H(v)-1$ edges to restore the connectedness. By induction, $K_{(k-1)p} - H'$ contains the complete p-partite graph $K_{k-1,\ldots,k-1}$. Since $N(v)$ is connected in H', all the vertices of $N(v)$ belong to the same vertex class of $K_{k-1,\ldots,k-1}$. Add v to the vertex class containing $N(v)$, and add one vertex in Q to each of the other vertex classes so that $K_{k,\ldots,k} \subseteq K_{kp} - H$. □

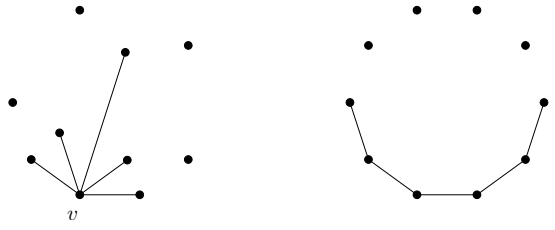


Figure 1: Two examples where removing n edges from the complete geometric graph on a set of $2n$ points disallows a non-crossing perfect matching.

Corollary 2 *For every complete geometric graph G on $2n$ vertices and for every subgraph H of G with at most $n-1$ edges, the geometric graph $G-H$ contains a non-crossing perfect matching. This bound is tight with respect to the cardinality of the set of forbidden edges.*

Proof. Apply the case $p = 2$ of Theorem 1, which states that $G-H$ contains a complete bipartite graph $K_{n,n}$. Color the point set according to this bipartition, say red and blue. This 2-colored point set has a non-crossing red-blue matching; that is, each edge of the matching connects a red and a blue point. Thus, this matching does not use edges of H.

Removing n edges from G does not always leave a non-crossing perfect matching, as can be seen in Figure 1 (left). There, if vertex v is matched to another point not using the drawn 'forbidden' edges, then this segment splits the point set into two sets of odd size, which disallows a non-crossing perfect matching. Thus, the bound of $n-1$ edges is tight. □

Another example that prohibits a non-crossing perfect matching without forbidden edges is shown in Figure 1 (right). In both examples the graph defined by the forbidden edges has one component that contains $n+1$ vertices. The size of the largest component in this graph turns out to be crucial for the existence of a non-crossing perfect matching without forbidden edges. To show this, we first show a result for colored point sets (which extends a known proof for 2-colored point sets).

Theorem 3 *Let S be a set of colored points in general position in the plane with $|S|$ even. Then S admits a non-crossing perfect matching such that every edge connects two points of distinct colors if and only if at most half the points in S have the same color.*

A related problem considering long alternating paths for multicoloured point sets was studied in [10].

Corollary 4 *For every complete geometric graph G on $2n$ vertices, and for every subgraph H of G with at most n vertices in each component, the geometric graph $G-H$ contains a non-crossing perfect matching.*

Note that Corollary 4 also implies Corollary 2.

Conjecture 1 *For every complete geometric graph G on a set of $2n$ points with $k \geq n-2$ of them in the interior of the convex hull and for every subgraph H of G which has at most $k+1$ edges, the geometric graph $G-H$ contains a non-crossing perfect matching.*

3 Non-crossing subtrees

In this section we investigate how many arbitrary edges can be removed from any complete geometric graph such that the remaining graph still contains a non-crossing tree of a given size. It turns out that the connectivity of the subgraph H defined by the removed edges is crucial for the size of the largest non-crossing subtree. We recall that the *connectivity* of a graph G is the size of a smallest vertex cut. A *vertex cut* of a connected graph G is a set of vertices whose removal disconnects G. A graph is called *k-connected* if its connectivity is k or greater.

Lemma 5 *For every complete geometric graph G on n vertices and for every subgraph H of G with connectivity k, the geometric graph $G-H$ contains a non-crossing subtree on $n-k$ vertices.*

In particular, Lemma 5 implies that for every subgraph H with $n-1$ edges of a complete geometric graph G on n vertices, the geometric graph $G-H$ contains a non-crossing subtree that spans $n-1$ vertices. Also, for every disconnected subgraph H of a complete geometric graph G, the geometric graph $G-H$ contains a non-crossing spanning tree.

Theorem 6 *For every $2 \leq k \leq n-1$, for every complete geometric graph G on n vertices, and for every subgraph H of G with at most $\lceil kn/2 \rceil$ edges, the geometric graph $G-H$ contains a non-crossing subtree that spans $n-k$ vertices. Moreover, the complete geometric graph G on n points in convex position has a subgraph H with $\lceil kn/2 \rceil$ edges such that $G-H$ has no non-crossing tree on $n-k+1$ vertices.*

Again we omit the proof and remark that for the convex complete geometric graph G considering as subgraph H the *Harary graph* $H_{k,n}$ [8], see Figure 2, yields the desired result.

We remark that also for point sets with many interior points we can not remove more edges than in the convex case to guarantee a non-crossing subtree of a given size in the remaining graph.

4 More classes of non-crossing geometric graphs

4.1 Spanning paths

Černý et al. [4] showed that for any subgraph $H = (V, E')$ of the convex complete geometric graph G on

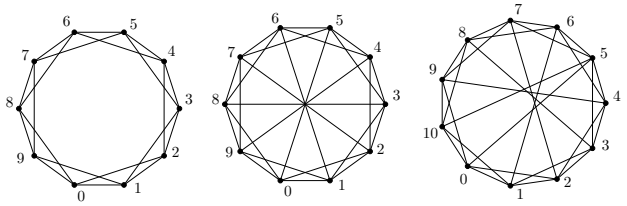


Figure 2: The Harary graphs $H_{4,10}$, $H_{5,10}$ and $H_{5,11}$.

n vertices with $|E'| \leq \lceil \frac{n}{2} \rceil - 1$, the geometric graph $G-H$ contains a non-crossing spanning path.

If the set S of n points has $k \leq \frac{n}{2}-2$ interior points, then we can not remove more than $\lceil \frac{n}{2} \rceil - 1$ edges of G; because each spanning path contains a perfect matching, for n even, and Figure 1 (left) shows that after removal of $\lceil \frac{n}{2} \rceil$ edges, the remaining graph does not even contain a non-crossing perfect matching.

Conjecture 2 *For every complete geometric graph G on a set of n points with $k \geq \lceil \frac{n}{2} \rceil - 2$ of them in the interior of the convex hull and for every subgraph H of G which has at most $k+1$ edges, the geometric graph $G-H$ contains a non-crossing spanning path.*

4.2 Spanning cycles

Point sets in convex position only admit one non-crossing spanning cycle. Therefore, removal of only one edge disallows such a cycle.

Conjecture 3 *For every complete geometric graph G on a set of n points with k of them in the interior of the convex hull and for every subgraph H of G which has at most $\lceil \frac{k}{2} \rceil$ edges, the geometric graph $G-H$ contains a non-crossing spanning cycle.*

Figure 3 shows an example where removal of $\lceil \frac{k+4}{2} \rceil$ edges disallows a non-crossing spanning cycle, for $k = n-3$.

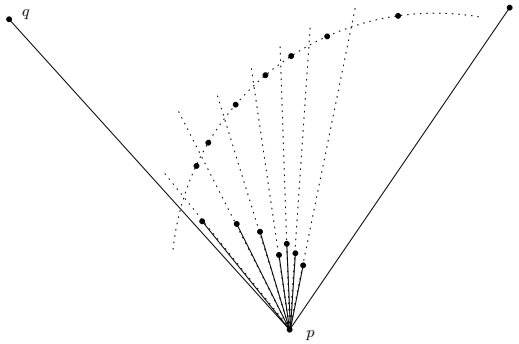


Figure 3: This point set contains no non-crossing spanning cycle if we disallow the $\lceil \frac{k+4}{2} \rceil$ drawn edges.

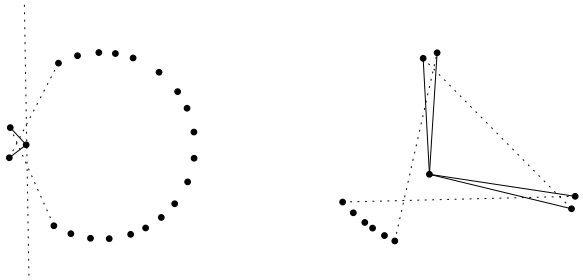


Figure 4: Removal of two avoidable edges disallows a triangulation (left) and removal of four avoidable edges disallows a pseudo-triangulation (right).

4.3 Triangulations and pseudo-triangulations

For each point set S there exist edges which appear in every triangulation of S, for example edges of the convex hull. We call these edges *unavoidable edges*. Edges which do not appear in every triangulation are called *avoidable edges*. Clearly, removal of only one unavoidable edge of the complete geometric graph on S disallows a triangulation for S. Thus, for removal we only consider avoidable edges.

Theorem 7 *For each subgraph $H = (V, E')$ of the complete geometric graph G on n vertices in convex position where E' is a set of at most $n - 3$ avoidable edges, the geometric graph $G - H$ contains a triangulation. This bound is tight with respect to the cardinality of E'.*

Interestingly, in the case of triangulations less (avoidable) edges can be removed if we also consider interior points. Figure 4 (left) shows an example.

Lemma 8 *There exist point sets with interior points, such that removal of two avoidable edges disallows a triangulation.*

We finally consider pseudo-triangulations. A *pseudo-triangle* is a simple polygon that has exactly three interior angles less than π. A *pseudo-triangulation* of a point set S is a partition of the convex hull of S into pseudo-triangles whose vertex set is exactly S. For the considered problem, pseudo-triangulations behave similar to triangulations. Note that for point sets in convex position triangulations and pseudo-triangulations coincide.

Lemma 9 *There exist point sets with interior points, such that removal of four avoidable edges disallows a pseudo-triangulation.*

Figure 4 (right) shows an example. The four solid edges are avoidable. To see that their removal disallows a pseudo-triangulation, observe that the face incident to the interior vertex with angle greater than π has to have at least four convex vertices, whereas a pseudo-triangle has exactly three.

References

[1] N. Alon, P. Erdős, *Disjoint edges in geometric graphs.* Discrete and Computational Geometry 4, pp. 287–290, 1989.

[2] V.I. Benediktovich, *Plane subgraphs in geometric complement of 2-factor and complete bipartite graph.* Electronic Notes in Discrete Mathematics 24, pp.31–38, 2006.

[3] J. Černý, *Geometric Graphs with No Three Disjoint Edges.* Discrete and Computational Geometry 34(4), pp. 679–695, 2005.

[4] J. Černý, Z. Dvořák, V. Jelínek, J. Kára, *Noncrossing Hamiltonian Paths in Geometric Graphs.* Discrete Applied Mathematics 155(9), pp. 1096-1105, 2007.

[5] P. Erdős, *On sets of distances of n points.* American Mathematical Monthly 53, pp. 248–250, 1946.

[6] S. Felsner, *Geometric Graphs and Arrangements: Some Chapters from Combinatorial Geometry.* Vieweg, Wiesbaden, 2004.

[7] W. Goddard, M. Katchalski, D.J. Kleitman, *Forcing Disjoint Segments in the Plane.* European Journal of Combinatorics 17(4), pp. 391–395, 1996.

[8] F. Harary, *The maximum connectivity of a graph.* Proceedings of the National Academy of Sciences of the United States of America 48, pp. 1142–1146, 1962.

[9] G. Károlyi, J. Pach, G. Tóth, *Ramsey-Type Results for Geometric Graphs, I.* Discrete and Computational Geometry 18, pp. 247–255, 1997.

[10] C. Merino, G. Salazar, J. Urrutia, *On the length of longest alternating paths for multicoloured point sets in convex position.* Discrete Mathematics 306, pp. 1791–1797, 2006.

[11] J. Pach, *Geometric graph theory.* In: J.D. Lamb, D.A. Preece, editors, Surveys in Combinatorics, 1999. London Mathematical Society Lecture Note Series 267, Cambridge University Press, pp. 167–200, 1999.

[12] J. Pach, J. Törőcsik, *Some geometric applications of Dilworth's theorem.* Discrete and Computational Geometry 12, pp. 1–7, 1994.

[13] G. Tóth, *Note on Geometric Graphs.* Journal of Combinatorial Theory, Series A 89(1), pp. 126–132, 2000.

[14] G. Tóth, P. Valtr, *Geometric Graphs with Few Disjoint Edges.* Discrete and Computational Geometry 22(4), pp. 633–642, 1999.

Computing the Dilation of Edge-Augmented Graphs in Metric Spaces

Christian Wulff-Nilsen*

Abstract

Let $G = (V, E)$ be an undirected graph with n vertices embedded in a metric space. We consider the problem of adding a shortcut edge in G that minimizes the dilation of the resulting graph. The fastest algorithm to date for this problem has $O(n^4)$ running time and uses $O(n^2)$ space. We show how to improve the running time to $O(n^3 \log n)$ while maintaining quadratic space requirement. In fact, our algorithm not only determines the best shortcut but computes the dilation of $G \cup \{(u, v)\}$ for every pair of distinct vertices u and v.

1 Introduction

In areas such as VLSI design, telecommunication, and distributed systems, a problem often arising is that of interconnecting a set of sites in a network of small cost. There are many different ways of measuring the cost of a network, such as its total length, minimum and maximum degree, diameter, and dilation (also known as stretch factor).

Spanners are sparse or economic representations of networks, making them important geometric structures in the areas mentioned above. They have received a great deal of attention in recent years, see e.g. surveys [2, 8].

A *t-spanner* is a graph embedded in a metric space such that, for any pair of vertices in this graph, the graph distance between them is at most t times their metric distance. The smallest t such that a geometric graph is a t-spanner is called the *dilation* of the graph.

Most algorithms construct networks from scratch, but frequently one is interested in extending an already given network with a number of edges such that the dilation of the resulting network is minimized.

Farshi et al. [7] considered the following problem: given a graph $G = (V, E)$ with n vertices embedded in a metric space, find a vertex pair $(u, v) \in V \times V$ (called a shortcut) such that the dilation of $G \cup \{(u, v)\}$ is minimized. They gave a trivial $O(n^4)$ time and $O(n^2)$ space algorithm for this problem together with various approximation algorithms.

In this paper, we present an $O(n^3 \log n)$ time and $O(n^2)$ space algorithm for the above problem. This algorithm not only computes the best shortcut but returns a table T with a row and a column for every vertex in G such that for any pair of distinct vertices u and v, $T(u, v)$ is the dilation of $G \cup \{(u, v)\}$.

The organization of the paper is as follows. In Section 2, we give various basic definitions and assumptions. In Section 3, we present one of the key ideas of the paper which gives a powerful way of obtaining the dilation of edge-augmented graphs. We present our algorithm and prove its correctness in Section 4 and in Section 5, we show that the above time and space bounds hold. In Section 6, we consider the case where the given graph is disconnected. Finally, we make some concluding remarks and pose open problems in Section 7.

2 Basic definitions and assumptions

Given a non-empty set M, we define a *metric* on M to be a function $d : M \times M \to \mathbb{R}_+$ which satisfies, for all $x, y, z \in M$,

$$d(x, y) = 0 \Leftrightarrow x = y$$
$$d(x, y) = d(y, x)$$
$$d(x, y) \leq d(x, z) + d(z, y)$$

The latter condition is known as the *triangle inequality*. The pair (M, d) is called a *metric space*.

Let $G = (V, E)$ be an undirected graph embedded in metric space (M, d) and assume that G is connected.

Given two vertices $u, v \in V$ which are connected by a path $P \subseteq E$ in G, we refer to P as a *shortest path* between u and v if $\sum_{e \in P} d(e)$ is minimal over all paths between u and v in G. We denote by $d_G(u, v)$ the length of such a path.

We define the *dilation* $\delta_G(u, v)$ of a pair of distinct vertices $u, v \in V$ as $d_G(u, v)/d(u, v)$. The dilation of G is defined as

$$\delta_G = \max_{u,v \in V, u \neq v} \delta_G(u, v).$$

In the following, G denotes an undirected, connected graph (V, E) embedded in metric space (M, d) and $n = |V|$ denotes the number of vertices of G.

3 Upper envelope functions

In this section, we consider certain upper envelope functions which will help us to compute the dilation of

*Department of Computer Science, University of Copenhagen, koolooz@diku.dk

graphs obtained from G by the addition of a shortcut (a single edge).

Let u, v, and w_1 be three fixed vertices of G such that $u \neq v$ and let w_2 be a fourth vertex of G (not fixed).

Let $G' = G \cup \{e\}$ be the graph obtained by adding shortcut $e = (w_1, w_2)$ to G. Suppose that $d_G(u, w_2) < d_G(u, w_1) + d(w_1, w_2)$. Then no shortest path in G' from u traverses e in the direction $w_1 \rightarrow w_2$. Letting $x = d_G(u, w_2) + d(w_2, w_1)$, $a = 1/d(u, v)$, $b = d_G(w_1, v)/d(u, v)$, and $c = \delta_G(u, v)$, we have

$$\delta_{G'}(u, v) = \min\{c, ax + b\} = \begin{cases} c & \text{if } x \geq \frac{c-b}{a} \\ ax + b & \text{if } x \leq \frac{c-b}{a}, \end{cases}$$

Observe that a, b, and c are constants, since u, v, and w_1 are fixed. Hence, the dilation between u and v in G' may be expressed as a piecewise linear function $\delta(x)$ of the length $x \geq 0$ of a shortest path among those paths in G' from u to w_1 having e as their last edge.

We refer to the graph of $\delta(x)$ as a *staircase step*, see Figure 1 (a). Assuming $(c - b)/a > 0$, the part of the graph on interval $[0, (c - b)/a]$ is a line segment with slope a, which we refer to as the *left leg* of $\delta(x)$.

If $(c - b)/a \leq 0$, we define the left leg of $\delta(x)$ to be the degenerate line segment with slope a starting and ending in the point $(0, \delta(0))$.

The part of the graph on interval $[\max\{0, (c - b)/a\}, \infty[$ is a horizontal halfline, called the *right leg* of $\delta(x)$.

We define the *slope* of $\delta(x)$ to be the slope a of its left leg.

The left and right leg of $\delta(x)$ meet in a single point. We refer to this point as the *tip* of $\delta(x)$.

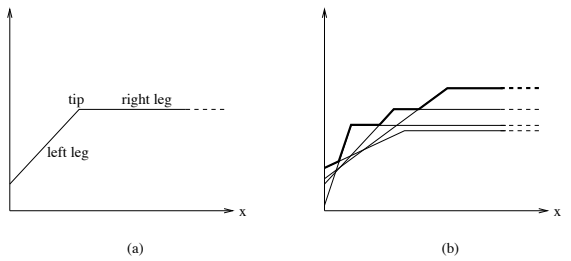


Figure 1: (a): A staircase step and (b): the upper envelope (thick line segments) of a set of four staircase steps.

Now, suppose we fix only u and w_1. For each $v \in V \setminus \{u\}$, we obtain a staircase step expressing the dilation between u and v in G'. We define $s_{(u,w_1)} : \mathbb{R}_+ \rightarrow \mathbb{R}_+$ to be the *staircase function* representing the upper envelope of the union of all these staircase steps as a function of x, see Figure 1 (b). Note that this function is piecewise linear and non-decreasing.

4 The algorithm and its correctness

In this section, we present our algorithm and prove its correctness.

Initially, $d_G(u, v)$ is computed and stored for each $(u, v) \in V \times V$, and a table T with an entry for each ordered pair of vertices of G is initialized. When the algorithm terminates, the following holds for each pair of distinct vertices (w_1, w_2)

$$\max\{T(w_1, w_2), T(w_2, w_1)\} = \delta_{G \cup \{(w_1, w_2)\}}. \qquad (1)$$

A subsequent step may update T in $\Theta(n^2)$ time such that $T(w_1, w_2) = \delta_{G \cup \{(w_1, w_2)\}}$ for all $w_1 \neq w_2$. A best shortcut is then a pair (w_1, w_2) maximizing $T(w_1, w_2)$.

The algorithm consists of a loop which iterates over all vertices of G. Let w_1 be the vertex in the current iteration. First, staircase functions $s_{(u,w_1)}$ are computed for each $u \in V$. Then for each vertex $w_2 \neq w_1$, entry (w_1, w_2) in T is set to

$$\begin{aligned} T(w_1, w_2) = \max\{s_{(u,w_1)}(d_G(u, w_2) + d(w_2, w_1)) \mid \\ u \in V, d_G(u, w_2) < d_G(u, w_1) + \\ d(w_1, w_2)\}. \end{aligned}$$

This is well-defined since $u = w_2$ satisfies $d_G(u, w_2) < d_G(u, w_1) + d(w_1, w_2)$.

The following theorem shows the correctness of our algorithm.

Theorem 1 *When the above algorithm terminates, (1) holds for each pair (w_1, w_2) of distinct vertices.*

Proof. Let (w_1, w_2) be any pair of distinct vertices of G and let $G' = G \cup \{(w_1, w_2)\}$. For any $u \in V$ for which $d_G(u, w_2) < d_G(u, w_1) + d(w_1, w_2)$ holds,

$$s_{(u,w_1)}(d_G(u, w_2) + d(w_2, w_1)) = \max_{v \in V \setminus \{u\}} \delta_{G'}(u, v).$$

Similarly, for any $u \in V$ for which $d_G(u, w_1) < d_G(u, w_2) + d(w_2, w_1)$ holds,

$$s_{(u,w_2)}(d_G(u, w_1) + d(w_1, w_2)) = \max_{v \in V \setminus \{u\}} \delta_{G'}(u, v).$$

Furthermore, for any $u \in V$, either $d_G(u, w_1) < d_G(u, w_2) + d(w_2, w_1)$ or $d_G(u, w_2) < d_G(u, w_1) + d(w_1, w_2)$ for otherwise,

$$\begin{aligned} d_G(u, w_2) + d(w_2, w_1) &\leq d_G(u, w_1) \\ &\leq d_G(u, w_2) - d(w_1, w_2) \\ &< d_G(u, w_2) + d(w_2, w_1), \end{aligned}$$

a contradiction. Thus, at termination,

$$\begin{aligned} \max\{T(w_1, w_2), T(w_2, w_1)\} &= \max_{u,v \in V, u \neq v} \delta_{G'}(u, v) \\ &= \delta_{G'}, \end{aligned}$$

as requested. □

5 Running time and space requirement

In this section, we show that the algorithm of the previous section has $O(n^3 \log n)$ running time and $O(n^2)$ space requirement. We will need the following lemma.

Lemma 2 *Given vertices u and w_1, the graph of staircase function $s_{(u,w_1)}$ consists of $O(n)$ line segments and one halfline and can be computed in $O(n \log n)$ time when $d_G(w_1, v)$ and $d_G(u, v)$ are precomputed for all $v \in V$. Furthermore, when this graph is given, $s_{(u,w_1)}(x)$ can be computed in $O(\log n)$ time for any $x \geq 0$.*

Proof. Note that when $d_G(w_1, v)$ and $d_G(u, v)$ are precomputed for all vertices v, each staircase step may be computed in constant time.

We represent the graph of $s_{(u,w_1)}$ as a polygonal chain P. To construct P, we start by computing each staircase step and the tip with maximum x-coordinate, say $x_{\max}$. The upper envelope of P to the right of $x_{\max}$ is the upper envelope of $O(n)$ horizontal halflines and may be computed in $O(n)$ time. The upper envelope of P on interval $[0, x_{\max}]$ is the upper envelope of $O(n)$ line segments. We use the algorithm of Hershberger [6] to compute this upper envelope in $O(n \log n)$ time. It follows that P may be constructed in $O(n \log n)$ time.

Clearly, P consists of line segments and exactly one halfline. We need to show that the number of line segments is $O(n)$.

Consider constructing P by iteratively adding staircase steps in non-decreasing order of slope. Let P_i be the upper envelope of the first i staircase steps.

Upper envelope P_1 consists of exactly one line segment (and one halfline). For $i > 1$, the left leg of the ith staircase step s_i intersects P_{i-1} at most once due to the order of staircase steps. Since the right leg is horizontal, it cannot intersect P_{i-1} more than once. Hence, P_i has at most two more line segments than P_{i-1}.

One fine point: a degeneracy may occur if the left leg of s_i overlaps with a line segment of P_{i-1}. It is easy to see that in this case, P_i cannot contain more line segments than P_{i-1}, again due to the order of the staircase steps.

Since there are $O(n)$ staircase steps, the above shows that P consists of $O(n)$ line segments.

Since $s_{(u,w_1)}$ is a non-decreasing function of x, we may apply a binary search in P to compute $s_{(u,w_1)}(x)$ for any $x \geq 0$. Since P consists of $O(n)$ line segments, this takes $O(\log n)$ time. □

We are now ready for the main result of this section.

Theorem 3 *The algorithm described in Section 4 has $O(n^3 \log n)$ running time and $O(n^2)$ space requirement.*

Proof. To prove the time bound, first observe that computing all-pairs shortest paths takes $O(n^3)$ time with the Floyd-Warshall algorithm [9] (faster algorithms exist [12] but they will not improve the asymptotic running time of our algorithm).

Furthermore, the graph of each staircase function is computed exactly once throughout the course of the algorithm. Hence, by Lemma 2, the total time spent on computing these functions is $O(n^3 \log n)$. Once the staircase functions have been computed, computing an entry of T takes $O(n \log n)$ time by Lemma 2. Since T has n^2 entries, computing T takes $O(n^3 \log n)$ time. When all entries in T have been computed, finding the best shortcut takes $O(n^2)$ time. Hence, the total running time of the algorithm is $O(n^3 \log n)$.

Space requirement is bounded by that of the Floyd-Warshall algorithm and the space for storing the staircase functions, the shortest path lengths, and the table T. The Floyd-Warshall algorithm requires $\Theta(n^2)$ space. Clearly, T and the shortest path lengths can be stored using a total of $\Theta(n^2)$ space. In each iteration of the algorithm, we only store n staircase functions. By Lemma 2, they take up a total of $O(n^2)$ space. □

6 Disconnected graph

Recall our assumption that G is connected. In this section, we show that some simple modifications of our algorithm allow us to handle the case where G is disconnected without affecting the worst-case running time and space requirement of the algorithm.

Note that if G consists of more than two connected components, G has infinite dilation and no single edge can be added to G to reduce the dilation, making the problem we consider trivial. Since there are efficient algorithms for determining the connected components of a graph, we may therefore restrict our attention to the case where G consists of exactly two connected components and assume that these two components have been computed.

So let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be the subgraphs defining the two connected components of G. For all shortcuts $(w_1, w_2) \in V_1 \times V_1 \cup V_2 \times V_2$, we set $T(w_1, w_2) = \infty$ since they leave the graph disconnected and hence leave the dilation of the graph unchanged.

As for the other entries in T, consider a pair of vertices (w_1, w_2) in $V_1 \times V_2$ and let $G' = G \cup \{(w_1, w_2)\}$. Let $u \neq v$ be two vertices of V. If $u, v \in V_1$ or $u, v \in V_2$ then clearly $\delta_{G'}(u, v) = \delta_G(u, v)$.

Now assume that $v \in V_1$ and $u \in V_2$. Then

$$\delta_{G'}(u, v) = ax + b,$$

where $x = d_G(u, w_2) + d(w_2, w_1)$, $a = 1/d(u, v)$, and $b = d_G(w_1, v)/d(u, v)$. Comparing this with the results of Section 3, we see that we in effect obtain staircase steps with no right leg. We let $s_{(u,w_1)}$ denote the

staircase function representing the upper envelope of the staircase steps obtained from each $v \in V_1$ as a function of x.

To determine all entries $T(w_1, w_2)$ of T where $(w_1, w_2) \in V_1 \times V_2$, we make the following small changes to the algorithm of Section 4. The loop only iterates over vertices $w_1 \in V_1$. Furthermore, we only compute staircase functions $s_{(u,w_1)}$ for $u \in V_2$ and we set

$$T(w_1, w_2) = \max\{\delta_{G_1}, \delta_{G_2}, \max\{s_{(u,w_1)}(d_G(u, w_2) + d(w_2, w_1)) | u \in V_2\}\}.$$

for each $w_2 \in V_2$.

By the above it follows that, at termination, the modified algorithm satisfies

$$\max\{T(w_1, w_2), T(w_2, w_1)\} = \delta_{G \cup \{(w_1,w_2)\}}$$

for all distinct pairs of vertices w_1 and w_2 in G.

Computing the graph of staircase function $s_{(u,w_1)}$ is done in $O(n \log n)$ time using the algorithm of Hershberger [6] (as in the proof of Lemma 2, we pick a maximum x-value in order to consider line segments instead of halflines. Pick, say, the largest x-value ever needed by the algorithm). The graph of $s_{(u,w_1)}$ has complexity $O(n)$ and when it is given, $s_{(u,w_1)}(x)$ can be computed in $O(\log n)$ time for any $x \geq 0$; the proof of these claims is similar to the proof of Lemma 2.

From the above and from the results of Section 5, it follows easily that all entries of T can be computed in $O(n^3 \log n)$ time using $O(n^2)$ space when G is disconnected.

7 Concluding remarks

We presented an $O(n^3 \log n)$ time and $O(n^2)$ space algorithm for the problem of computing the best shortcut of a geometric graph $G = (V, E)$ with n vertices. This improves upon a previous bound of $O(n^4)$ time and $O(n^2)$ space [7]. Our algorithm in fact solves a harder problem, namely that of computing the dilation of $G \cup \{(u, v)\}$ for each pair of distinct vertices u and v.

The problem stated in [7] of whether there exists a linear space algorithm with $o(n^4)$ running time for finding the best shortcut of a geometric graph remains open. We pose the following problems. Is our algorithm optimal in terms of running time? Can we reduce space requirement without affecting running time? Is it possible to extend our results to the more general case of adding a constant number of edges to G?

Acknowledgments

I thank Pawel Winter and Martin Zachariasen for their comments and remarks.

References

[1] A. Czumaj and H. Zhao. *Fault-Tolerant Geometric Spanners.* Discrete Comput Geom 32:207–230 (2004).

[2] D. Eppstein. *Spanning trees and spanners.* In J.-R. Sack and J. Urrutia, editors, Handbook of Computational Geometry, pages 425–461, Elsevier Science Publishers, Amsterdam, 2000.

[3] G. Narasimhan and M. Smid. *Approximating the stretch factor of Euclidean graphs.* SIAM J. Comput. 30 (3) (2000), 978–989.

[4] G. Rote. *Computing the minimum Hausdorff distance between two point sets on a line under translation.* Information Processing Letters 38 (1991), 123–127.

[5] J. Gudmundsson, G. Narasimhan, and M. Smid. *Fast pruning of geometric spanners.* STACS 2005:508–520.

[6] J. Hershberger. *Finding the upper envelope of n line segments in $O(n \log n)$ time.* Information Processing Letters, Vol. 33, no. 4, 1989, pp. 169–174.

[7] M. Farshi, P. Giannopoulos, and J. Gudmundsson. *Finding the Best Shortcut in a Geometric Network.* 21st Ann. ACM Symp. Comput. Geom. (2005), pp. 327–335.

[8] M. Smid. *Closest point problems in computational geometry.* In J.-R. Sack and J. Urrutia, editors, Handbook of Computational Geometry, pages 877–935, Elsevier Science Publishers, Amsterdam, 2000.

[9] R. W. Floyd. *Algorithm* 97 *(SHORTEST PATH).* Communications of the ACM, 5(6):345, 1962.

[10] S. Arya, G. Das, D. M. Mount, J. S. Salowe, and M. Smid. *Euclidean spanners: short, thin, and lanky.* Proc. 27th ACM STOC, 1995, pp. 489–498.

[11] S. Langerman, P. Morin, and M. Soss. *Computing the maximum detour and spanning ratio of planar chains, trees and cycles.* Proceedings of the 19th International Symposium on Theoretical Aspects of Computer Science (STACS '02), Lecture Notes in Computer Science, Vol. 2285, 2002, pp. 250–261.

[12] T. M. Chan. *More algorithms for all-pairs shortest paths in weighted graphs.* Proceedings of the thirty-ninth annual ACM symposium on Theory of computing (2007), pp. 590–598.

[13] X. Y. Li and Y. Wang. *Efficient construction of low weighted bounded degree planar spanner.* Int. J. Comput. Geometry Appl. 14(1–2):69–84 (2004).

Approximating the Minimum Spanning Tree of Set of Points in the Hausdorff Metric

Victor Alvarez* Raimund Seidel†

Abstract

We study the problem of approximating $\mathrm{MST}(P)$, the Euclidean minimum spanning tree of a set P of n points in $[0,1]^d$, by a spanning tree of some subset $Q \subset P$. We show that if the *weight* of $\mathrm{MST}(P)$ is to be approximated, then in general Q must be large. If the *shape* of $\mathrm{MST}(P)$ is to be approximated, then this is always possible with a small Q.

More specifically, for any $0 < \varepsilon < 1$ we prove:

(i) There are sets $P \subset [0,1]^d$ of arbitrarily large size n with the property that any subset $Q' \subset P$ that admits a spanning tree T' with $\big||T'| - |\mathrm{MST}(P)|\big| < \varepsilon \cdot |\mathrm{MST}(P)|$ must have size at least $\Omega(n^{1-1/d})$. (Here $|T|$ denotes the weight, i.e. the sum of the edge lengths of tree T.)

(ii) For any $P \subset [0,1]^d$ of size n there exists a subset $Q \subseteq P$ of size $O(1/\varepsilon^d)$ that admits a spanning tree T that is ε-close to $\mathrm{MST}(P)$ in terms of Hausdorff distance (which measures shape dissimilarity).

(iii) This set Q and this spanning tree T can be computed in time $O(\tau_d(n) + 1/\varepsilon^d \log(1/\varepsilon^d))$ for any fixed dimension d. Here $\tau_d(n)$ denotes the time necessary to compute the minimum spanning tree of n points in $\mathbb{R}^d$, which is known to be $O(n \log n)$ for $d = 2$, $O((n \log n)^{4/3})$ for $d = 3$, and $O(n^{2-2/(\lceil d/2 \rceil + 1) + \phi})$, with $\phi > 0$ arbitrarily small, for $d > 3$ (see [1]).

All the results hold not only for the Euclidean metric L_2 but also for any L_p metric with $1 \leq p \leq \infty$ as underlying metric.

1 Introduction

The approximation of geometric problems by means of reducing the size of the input has been the subject of study of many researchers. The idea is the fast identification of the part of the input that matters for the problem at hand and the use of this extracted data to speed up the computations.

In [2], Agarwal *et al.* developed a framework, called Coresets, to approximate extent measures of a given set of points P in any fixed dimension d. Such extent measures include the diameter, the width, the radius of the minimum enclosing cylinder, etc. Their idea is basically the computation of a subset P' of P whose size depends exclusively on ε and d and, whose convex hull approximates the convex hull of P. Then, use this new convex hull for further computations and argue that this produces good approximations for the desired extent measures.

In this paper we are interested in approximating the Euclidean minimum spanning tree of a set $P \subset \mathbb{R}^d$ of points, but not in the sense of, say, Clarkson [4], who wants to quickly find some spanning tree of P whose weight is close to that of $\mathrm{MST}(P)$. We are instead interested in finding a spanning tree of a *small subset* of P that in some sense approximates $\mathrm{MST}(P)$. We will show that the core set approach outlined above cannot work in this context if the approximation measure is the weight of the trees. However, if we want to approximate $\mathrm{MST}(P)$ in a more topological (or shape) sense, then this is indeed possible using a spanning tree of a subset of P whose size depends exclusively on ε, the approximation parameter, and on d. This result potentially has applications in Image Comparison and Pattern Recognition.

Throughout the paper let $0 < \varepsilon < 1$ be a fixed constant. Also the dimension d is meant to be fixed.

2 MST(P) admits no constant size subset approximation with respect to weight

The goal of this section is to prove the following result:

Theorem 1 *For each $n = k^d$ with $k \in \mathbb{N}$ there exists a set $P \subset [0,1]^d$ of n points such that any subset Q' of P that admits a spanning tree T' with $|T'| \geq (1-\varepsilon)|MST(P)|$ must have size at least $\Omega(k^{(d-1)})$.*

Note that this theorem clearly implies Claim (i) of the abstract.

Proof. Let $n = k^d$ with $k \in \mathbb{N}$ and let $\mathbb{G}^d$ be the d-dimensional grid over $[0,1]^d$ of cell size $\delta = 1/(k-1)$. Let P be the set consisting of the grid points of $\mathbb{G}^d$. It is clear that $|P| = n$. Any Euclidean minimum spanning tree of such a set P only contains grid edges. Thus $|\mathrm{MST}(P)| = (n-1) \cdot \delta = (n-1)/(k-1) > n/k$. See Figure 1.

Now let T' be a spanning tree of some $Q' \subset P$ such that $|T'| \geq (1-\varepsilon)|\mathrm{MST}(P)|$. Every edge inside

*International Max-Planck Research School for Computer Science and Fachrichtung Informatik, Universität des Saarlandes, alvarez@mpi-inf.mpg.de

†Fachrichtung Informatik, Universität des Saarlandes, rseidel@cs.uni-sb.de

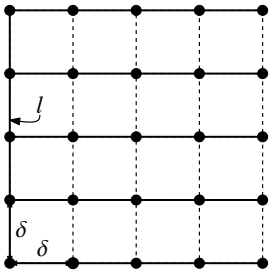


Figure 1: For $P \subset [0,1]^2$ of size $25 = 5^2$, a rectilinear spanning tree is shown. The vertical line l works as a backbone and connecting all the horizontal lines of $\mathbb{G}^2$ to it gives us a total weight of exactly $24/4 = 6$.

the unit cube $[0,1]^d$ has length at most $\sqrt{d}$. Hence $|T'| < |Q'| \cdot \sqrt{d}$. Combining this last inequality with the ones above we have

$$|Q'| \cdot \sqrt{d} > |T'| \geq (1-\varepsilon)|\mathrm{MST}(P)| > (1-\varepsilon)\frac{n}{k}\,.$$

Since $n = k^d$ and ε and d are constant, the result follows.

□

3 The Hausdorff metric

The Hausdorff metric allows to define distances between subsets of a metric space. In our case the metric space is $\mathbb{R}^d$ with the usual Euclidean metric.

Definition 1 (Hausdorff distance) *The* Hausdorff distance $H(A,B)$ *between two non-empty subsets A, B of $\mathbb{R}^d$ is defined to be the radius of the largest open ball centered in one set and not meeting the other set.*
We say that A and B are ε-close *iff $H(A,B) \leq \varepsilon$.*

It is well known that the Hausdorff distance constitutes a metric on the space of all non-empty compact subsets of $\mathbb{R}^d$. Moreover, in a way it expresses the shape similarity, or rather dissimilarity between sets: $H(A,B) = 0$ means A and B must be the same, i.e. they are not at all dissimilar, and A and B are ε-close means that they are only ε-dissimilar in the sense that for any point in one set within Euclidean distance ε there must be a point of the other set. Many computational geometry papers have used the Hausdorff distance as a measure of similarity/dissimilarity between subsets of $\mathbb{R}^d$, see e.g. [3]. We will use the Hausdorff distance to measure similarity/dissimilarity between spanning trees of finite sets embedded in $\mathbb{R}^d$, where such a tree is considered a subset of $\mathbb{R}^d$, namely the union of the segments formed by its edges.

It will turn out that if instead of closeness in weight we consider closeness in Hausdorff distance the Euclidean minimum spanning tree of any finite $P \subset \mathbb{R}^d$ admits a good approximation by a spanning tree of a constant sized subset of P.

4 Approximating MST(P) by shape

At first a few graph theoretic preliminaries.

Let G be a complete undirected graph with vertex set P and with weighted edges. For the sake of exposition we assume that all edge weights are distinct, and thus the minimum edge of any cut of G and also the minimum spanning tree $\mathrm{MST}(P)$ are unique. This assumption can be justified using a standard perturbation argument. Let $\overline{P} = \langle P_1, \ldots, P_k \rangle$ be a partition of P into $k \geq 2$ non-empty "clusters," and let $\overline{G}$ be the graph obtained from G by contracting each cluster in $\overline{P}$ into a single node. $\overline{G}$ has parallel edges and self-loops, still, its minimum spanning tree $\mathrm{MST}(\overline{P})$ is unique. Consider the forest on P formed by the $k-1$ edges of G that induce the edges of $\mathrm{MST}(\overline{P})$. Let us call this forest the *minimum cluster forest of P with respect to $\overline{P}$*, for short $\mathrm{MCF}(P, \overline{P})$.

What is the relationship between the edges in $\mathrm{MCF}(P, \overline{P})$ and $\mathrm{MST}(P)$?

Lemma 2 *Every edge in $\mathrm{MCF}(P, \overline{P})$ also is an edge of $\mathrm{MST}(P)$.*

Proof. Let e be an edge of $\mathrm{MCF}(P, \overline{P})$ and let $\overline{e}$ be the corresponding edge of $\mathrm{MST}(\overline{P})$. The removal of $\overline{e}$ from $\mathrm{MST}(\overline{P})$ results in two subtrees producing a partition of the node set $\overline{P}$ into two sets $\overline{R}$ and $\overline{S}$. The edge $\overline{e}$ must be the shortest edge between nodes (i.e. clusters) in $\overline{R}$ and in $\overline{S}$ and hence e must be the shortest edge between (original) vertices in $R = \bigcup \overline{R}$ and in $S = \bigcup \overline{S}$. Since R and S form a partition of P this means that e must be an edge of $\mathrm{MST}(P)$. □

Let us call an edge of G *long* (with respect to $\overline{P}$) iff it is longer than any edge connecting two vertices in the same cluster of $\overline{P}$.

Lemma 3 *Every long edge of $\mathrm{MST}(P)$ is also an edge of $\mathrm{MCF}(P, \overline{P})$.*

Proof. Let e be a long edge in $\mathrm{MST}(P)$. Similar to the previous proof the edge e induces a partition of P into R and S, and e is the shortest edge connecting vertices in R with vertices in S. No cluster of $\overline{P}$ can have a vertex both in R and in S, since such two vertices would be connected by an edge shorter than the long edge e, a contradiction to e being the shortest edge between R and S. Thus R and S induce a partition of the cluster set $\overline{P}$ into $\overline{R}$ and $\overline{S}$, and $\overline{e}$ (induced by e) is the shortest edge connecting a cluster in $\overline{R}$ with a cluster in $\overline{S}$. Thus $\overline{e}$ is an edge of $\mathrm{MST}(\overline{P})$ and therefore e is an edge of $\mathrm{MCF}(P, \overline{P})$. □

In the following P will a set of points in $\mathbb{R}^d$ and the weight of the edge connecting two points $x, y \in P$ will be the Euclidean distance between x and y. We are now able to present the main result of this section which will prove Claim (ii) of the abstract.

Theorem 4 *Let P be a set of points in $[0,1]^d$ and let $0 < \varepsilon < 1$ be a given parameter. It is possible to find a spanning tree T of some subset Q of P such that MST(P) and T are ε-close and $|Q| = O(1/\varepsilon^d)$.*

Proof. We will start by imposing a d-dimensional grid $\mathbb{G}^d$ of cell size $\delta = \frac{2\varepsilon}{3\sqrt{d}}$ over P. The grid $\mathbb{G}^d$ induces a partition $\overline{P}$ of P into $k = O(1/\varepsilon^d)$ clusters, with each cluster begin composed of the set of points contained in a cell of $\mathbb{G}^d$. See Figure 2. Note that two points in the same cluster are at most $2\varepsilon/3$ apart.

The claimed set Q will be the points in P incident to the edges of the minimum cluster forest MCF$(P, \overline{P})$. Since there are $k-1$ edges in MCF$(P, \overline{P})$ it follows that $|Q| = O(1/\varepsilon^d)$.

The claimed spanning tree T of Q will contain all edges in MCF$(P, \overline{P})$ and in addition for each cluster C in $\overline{P}$ an arbitrary spanning tree of the points of Q in C. See Figure 2.

We claim that T and MST(P) are ε-close.

We need to prove that for every point on T there is a point on MST(P) within distance at most ε, and vice versa.

Let e be an edge of T. If e is an edge of MCF$(P, \overline{P})$, then by Lemma 2 it is also an edge of MST(P) and thus every point x on e is within distance $0 < \varepsilon$ of some point of MST(P). If e is an edge connecting two points of the same cluster, then its length is at most $2\varepsilon/3$. Thus any point x on e is at most at distance $\varepsilon/3 < \varepsilon$ from one of e's endpoints, which are both in MST(P).

Now let e be an edge of MST(P). If it has length bigger than $2\varepsilon/3$, then it is long in the sense of Lemma 3, and therefore it is contained in MCF$(P, \overline{P})$ and hence also in T. Thus every point x on e is within distance $0 < \varepsilon$ of some pont of T. If e has length less than $2\varepsilon/3$, then every point x is within distance $\varepsilon/3$ of an endpoint v of e. Let q some point of Q in the cluster containing v. The distance between v and q is at most $2\varepsilon/3$, and thus by the triangle inequality the distance between x and q (which lies on T) is at most ε. □

This result says that it is possible to find a constant-size subset Q of P along with a spanning tree T of Q such that shape-wise T and MST(P) look essentially the same. This gives a method to sort of "compress" MST(P) to a tree that is close in shape but has constant size. Note, however, that one cannot conclude anything from T about the total weight |MST(P)|.

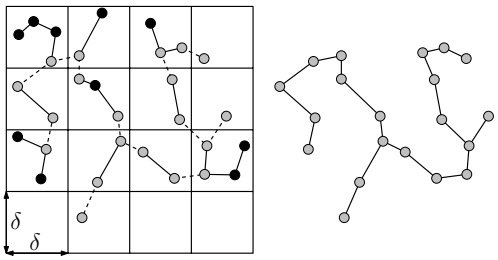


Figure 2: The points chosen to form Q are highlighted in light gray. The dashed edges connect points in different clusters of P induced by $\mathbb{G}^2$

5 Computing T

The only computationally non-trivial step in computing Q and T is the determination of the edges of the cluster forest MCF$(P, \overline{P})$. The straigtforward way of computing these edges, forming the cluster graph $\overline{G}$ and computing its minimum spanning tree MST$(\overline{P})$, leads to an $\Theta(n^2)$ time algorithm in the worst case, since $\overline{G}$ can have $\Theta(n^2)$ non-loop edges. (We assume here ε and d to be fixed.)

Lemma 2 implies that for computing MST$(\overline{P})$ it suffices to consider only those edges that are induced by edges of MST(P). This suggests the following algorithm: Compute $B =$ MST(P), for each grid imposed cluster contract the edges of B within the cluster to produce a contracted graph $\overline{B}$. Compute the minimum spanning tree of $\overline{B}$, which by Lemma 2 is the same as MST$(\overline{P})$.

If $\tau_d(n)$ denotes the time necessary to compute the Euclidean minimum spanning tree of n points in $\mathbb{R}^d$, then the time necessary for the outlined method is $\tau_d(n)$ for computing MST(P), plus $O(n)$ for computing $\overline{B}$ and $O(n+N \log N)$ for computing the minimum spanning tree of $\overline{B}$, where $N = O(1/\varepsilon^d)$ is the number of occupied grid cells, which is constant if ε and d are considered to be constant. The total time for the whole method is then dominated by $\tau_d(n)$, which is known to be $O(n \log n)$ for $d = 2$ and $O((n \log n)^{4/3})$ for $d = 3$, and $O(n^{2-2/(\lceil d/2 \rceil + 1) + \phi})$, with $\phi > 0$ arbitrarily small, for $d > 3$ (see [1]).

Other methods suggest themselves, but they are either incorrect or do not seem to lead to better time bounds. For instance, we could choose a small sample set of points from each occupied cluster and compute the minimum spanning tree of the union of these sample sets. However, the tree produced this way may be very different in shape from MST$(\overline{P})$ and will not lead to a tree that is ε-close to MST(P). Or, we could run a minimum spanning tree algorithm on the clusters (without forming $\overline{G}$ or some subgraph explicitly) by repeatedly solving so-called bi-chromatic closest pair problems. However, this is unlikely to produce a better running time, since the complexity of solving a

bi-chromatic closest pair problem on n points in $\mathbb{R}^d$ is known to be $\Theta(\tau_d(n))$, see [5].

Finding a faster algorithm for computing a constant sized tree that is ε-close to MST(P) looks like a challenging problem.

6 Conclusion

We have shown that in general it is not possible to approximate well the weight of the Euclidean minimum spanning tree of a set of points P in $\mathbb{R}^d$ with a subset of size independent of the size of P. However, changing the notion of approximation, we have shown, that it is possible to compute a spanning tree T of some small subset $Q \subseteq P$ such that the Hausdorff distance between T and the minimum spanning tree of P is small, which means that the two trees are very similar in shape. This potentially has applications in Image Comparison or Pattern Recognition, and also provides a potential way of compressing MST(P) in a meaningful and interesting way.

Our results and methods apply not just to the standard Euclidean L_2 metric but also to any L_p metric for $1 \leq p \leq \infty$.

Acknowledgments

The first author would like to thank the International Max-Planck Reseach School for Computer Science for the financial support during the development of this work.

References

[1] Pankaj K. Agarwal, Herbert Edelsbrunner, Otfried Schwarzkopf, and Emo Welzl. *Euclidean Minimum Spanning Trees and Bichromatic Closest Pairs.* Discrete Comput. Geom. 6(5):407-422, 1991.

[2] Pankaj K. Agarwal, Sariel Har-Peled, and Kasturi R. Varadarajan. *Approximating Extent Measures of Points.* J. ACM, 51(4):606-635, 2004.

[3] H. Alt and L.J. Guibas. *Discrete Geometric Shapes: Matching, Interpolation, and Approximation* In J.R. Sack and J. Urrutia, editors, **Handbook of Computational Geometry.** Elsevier Science Publishers B.V. North-Holland, Amsterdam, pp. 121–153, 2000.

[4] Kenneth L. Clarkson *Fast Expected-time and Approximation Algorithms for Geometric Minimum Spanning Trees.* In STOC '84: Proceedings of the 16th Annual ACM Symposium on Theory of Computing, 342-348, 1984.

[5] Drago Krznaric, Christos Levcopoulos, and Bengt J. Nilsson. *Minimum Spanning Trees in d Dimensions.* Nordic J. of Computing, 6(4):446-461, 1999.

[6] Andrew C. Yao. *On Constructing Minimum Spanning Trees in k-dimensional Spaces and Related Problems.* SIAM Journal on Computing, 11(4):721-736, 1982.

Optimization Techniques for Geometry Processing

Pierre Alliez

GEOMETRICA project-team
INRIA Sophia-Antipolis, BP 93
06902 Sophia-Antipolis cedex, France

e-mail: pierre.alliez@sophia.inria.fr
URL: http://www-sop.inria.fr/geometrica/team/Pierre.Alliez/

Abstract

Mesh generation, surface reconstruction and quadrangle surface tiling are common problems in digital geometry processing. My favourite approach consists of designing an energy function for each problem, such that good solutions correspond to low-energy ones. The desired solutions can then be found by applying optimization techniques. This talk will put an emphasis on the variety of the designed energy functions and of the associated minimization techniques. The energy functions range from size and compactness of Voronoi cells to alignment of gradients of implicit functions through size and alignment of quadrangle edges. The optimization techniques range from 4D function approximation to 3D generalized eigenvalue problems through computation of harmonic one-forms on triangle surface meshes. Each of the algorithms presented will be illustrated with live demos implemented with the CGAL library.

Geometry with Imprecise Lines*

Maarten Löffler† Marc van Kreveld†

Abstract

Practical application of geometric algorithms is hindered by data imprecision. One of the primitive elements in geometry is the concept of a line. We investigate what is the right way to model imprecise lines, and present algorithms to compute bounds on the solution to linear programming or vertical extent problems on a set of imprecise lines.

1 Introduction

Data imprecision constitutes an important gap between theory and practice in computational geometry. We have studied the effects of imprecision on points in the plane [12], and given algorithms to infer bounds on basic measures of point sets from the imprecision information. The two most basic elements in Euclidean geometry are points and lines. This paper is an investigation into the meaning of lines in an imprecise context, and what can be done with them algorithmically. In the next section we give a definition of *imprecise lines*, and in Sections 3 and 4 we give algorithms to infer bounds on the outcome of two important and well studied problems on lines: linear programming and vertical extent.

2 Imprecise lines

An imprecise line is a line of which we don't know exactly where it is, but we do have some information. The easiest way to model this is to define a set of *candidate* lines for our imprecise lines: a set L of lines in $\mathbb{R}^2$. Such a set will not be finite in most interesting cases. We will assume that we have a collection of imprecise lines $\mathcal{L}$: a set of sets of lines in the plane. Our goal is to do the same computations on a set of imprecise lines that we could do on a set of normal lines. To be able to do so, we must put some more restrictions on the sets.

An imprecise line L is the set of possible lines that an unknown line l could be. We would like to model this set in a natural way. When treating imprecise points, we restricted the regions to be *connected*, and furthermore to be *convex*. Additional constraints to make the regions easier to handle may include *constant description size*, or *linear* (polygonal) boundaries. We would like to have similar properties for imprecise lines.

We will make a clear distinction between lines and *directed* lines. Problems that take a set of lines as input usually treat them as either directed or undirected lines. There is also an important topological difference between the set of all lines in $\mathbb{R}^2$ and the set of all directed lines in $\mathbb{R}^2$: the former is isomorphic to the Möbius-strip, while the latter is isomorphic to a cylinder.

Connectedness is easy to define for a set of lines L: if two lines can be transformed into each other by a continuous movement inside L, they are connected.

Convexity of sets of lines is harder to define. This subject has been studied by various people, and several different definitions have been proposed. A first approach is to define convexity based on point-line duality [2] and convex point sets. The straightforward way to do this has the drawback that vertical lines cannot be represented. Rosenfeld [11] proposes a definition that gets around this problem and has nice properties, but which is not translation-invariant. Goodman [6] argues that no natural definition can exist, and gives a definition that drops the connectedness of convex sets. Gates [5] defines convexity for sets of *directed* lines in a natural way. Bhattacharya and Rosenfeld [1] give an extensive comparison between the various definitions.

With the idea in mind that we want to represent imprecise lines with our convex sets, it seems reasonable to assume that for a given imprecise line, there is at least one direction in which the line certainly does *not* lie. This need not be the same for all lines, so not allowing vertical lines is a bit too restrictive, but if we assume that for every imprecise line there is some forbidden direction, we can use the following quite natural definition for convexity (for both directed and undirected lines):

Definition (*convex*)**:** We will call a set of lines L *convex* if there exists a direction d such that no line $l \in L$ lies in direction d, and for any pair of lines $l, m \in L$ the following holds:

*Partially supported by the Netherlands Organisation for Scientific Research (NWO) through the project GOGO.

†Department of Information and Computing Sciences, Utrecht University. {loffler,marc}@cs.uu.nl

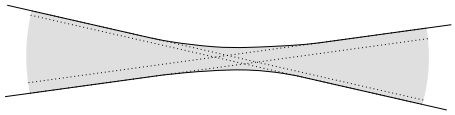


Figure 1: A convex set of lines.

- If l and m are parallel, then all other lines parallel to them and between them are also in L.
- If l and m intersect in a point p, then all other lines through p with a direction between l and m, rotating such that d is not encountered, are also in L.

Let L be a convex set of lines, and sort their directions cyclicly from d to d. Then there are two lines $l, m \in L$ that have the smallest and largest direction. Let the angle between l and m be α. We call α the *limit angle* of L. If the lines are undirected, α is smaller than π. If the lines are directed, α must be smaller than 2π. For undirected lines, this means L will either be a strip ($\alpha = 0$), or it can be described by two convex curves that have asymptotes with an angle α between them, see Figure 1. The set L is the set of all lines that lie completely within the portion of the plane bounded by the curves.

For directed lines, something interesting happens as soon as $\alpha > \pi$. We can use the convexity properties to prove that if some line l is in L, any translation of l that is rotated some arbitrarily small ε is also in L. However, when talking about imprecise lines it seems reasonable to assume that $\alpha < \pi$. In that case, our definition for directed lines also coincides with the definition from [5]. Our definition of convexity also coincides with the dual definition, when we rotate the plane such that d becomes vertical.

The other properties we mentioned, constant description size and piecewise linear boundaries, are now also easy and natural to define for convex sets of lines: if the defining curves are of constant description size or polygonal, then so is the set of lines.

If a set of lines is connected, convex with a limit angle $\alpha < \pi$, and has linear boundaries and constant description size, we call it a *bundle*. There are of course also different possibilities to naturally describe imprecise lines. Another natural model would be the set of lines stabbing two given regions (imprecise points) in order. These line-sets will in general not be convex. Imprecise linear programming also occurs in other fields, for example in biology [8]. These uses suggest the *axis-model*: an imprecise line is the collection of lines that intersect the x- and y-axes in certain fixed intervals. This model clearly fits within the stabbing model, and if no interval contains the origin it also fits within our description of convexity.

3 Linear programming

Linear programming is a widely studied problem where you are given a set of directed lines, and want to find the point with the lowest y-coordinate, under the restriction that it lies to the left of all lines. It can be solved in $O(n)$ time. In an imprecise context, we are interested in finding the set of lines in a collection of bundles $\mathcal{L}$ that maximises or minimises this value, since this gives us bounds on the possible values.

3.1 Largest value

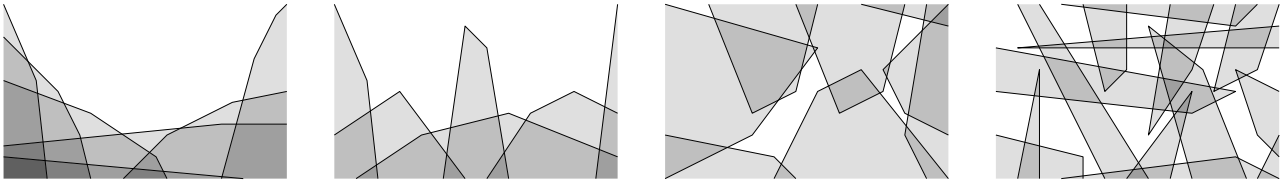
We want to choose one directed line from every bundle such that the point to the left of all lines with the lowest y-coordinate is as high as possible. To solve this problem, consider for some imprecise line L the line segments that make up the left boundary of its bundle. Take the lines supporting those segments, and do this for all imprecise lines. Apply a classical linear time algorithm for linear programming to the resulting set of lines. It is not hard to prove that the solution we find this way is the solution to the imprecise problem; we defer the proof to the full version.

3.2 Smallest value

Now we want to find the lowest point such that in every bundle, there is a line that has this point on its left. Therefore, we can define for each imprecise line L the *potential free space* as the union of the half-planes to the left of all lines in L. Next we define the potential free space of a collection of imprecise lines $\mathcal{L}$ as the intersection of the potential free spaces of all imprecise lines $L \in \mathcal{L}$. If p is a point in the plane, a choice of lines for $\mathcal{L}$ that has p to the left of all of them exists if and only if p lies in the potential free space of $\mathcal{L}$.

As a consequence, we are now looking for the lowest point in the intersection of a collection of concave regions in the plane. We could find this point by explicitly computing this intersection, which takes $O(n^2)$ time in general, and in Section 3.2 we show that in general this is indeed the best we can do. However, when our collection of imprecise lines satisfies some additional, natural constraints, we can in some cases solve the problem more efficiently. We call a bundle *diagonal* if it contains no horizontal or vertical lines. We call a bundle *upfacing* if all lines in the bundle are oriented from left to right. We will call a bundle c-fat if $\alpha < \pi - c$ for some constant $c > 0$.

Diagonal bundles. Let $\mathcal{L}$ be a collection of diagonal and upfacing bundles, see Figure 2(a). We can solve



(a) (b) (c) (d)

Figure 2: (a) The potential free space (white) of a set of diagonal, upfacing bundles. (b) The potential free space of a set of upfacing bundles. (c) The potential free space of a set of fat bundles. (d) The potential free space of a general set of bundles.

the problem in linear expected time using standard randomised incremental construction techniques. The details are in the full paper.

Upfacing bundles. Let $\mathcal{L}$ be a collection of upfacing bundles, see Figure 2(b). We can solve the problem by computing the upper envelope of the potential free regions in $O(n \log n)$ time [7]. The complexity of this envelope is $O(n\alpha(n))$, and we can clearly find the lowest point on it by just looking at all the points.

This bound is optimal, since we can reduce the *maximum gap* problem to it, which has a $\Theta(n \log n)$ lower bound [9]. Given a set of real numbers, for each number we create a wedge with the number as the x-coordinate of its top, and some constant y-coordinate. The wedges have two halflines with slope 1 and -1, going down. The lowest point above all wedges corresponds to the two consecutive numbers with the largest difference.

Fat bundles. Fatness seems to be a very reasonable restriction on imprecise lines. We already assumed that for each imprecise line, there is some direction d which we know the line does *not* have. We now weaken this a bit further, by saying that there is at least a small angle of directions the line cannot have.

Let $\mathcal{L}$ be a collection of fat bundles, see Figure 2(c). Efrat *et al.* [3] prove that the union of a set of δ-fat wedges can be computed in $O(n \log n)$ time. The proof can be adapted to also work for our more general bundles instead of wedges.

General bundles. Let $\mathcal{L}$ be a collection of bundles without any further restrictions, see Figure 2(d). In this case, it is likely that there is no algorithm that solves the problem faster than in $\Theta(n^2)$ time. We can reduce from covering a rectangle with strips, which falls in a class of $O(n^2)$ problems [4]. It is easy to achieve quadratic time by just explicitly computing the arrangement of the bundles.

4 Vertical extent

The *vertical extent* of a set of lines is the shortest vertical line segment that intersects a given set of lines. This problem is dual to finding the smallest vertical distance between two parallel lines containing a set of points. It is an LP-type problem [10], and can therefore also be solved in $O(n)$ time. When the lines are imprecise, we are again interested in the smallest and largest possible values for this problem.

4.1 Largest value

We call a bundle *vertical* if it contains at least one vertical line. When there are vertical bundles, the problem is trivial and the answer is ∞, since we can take two parallel lines that are arbitrarily close to vertical, and the vertical distance between them will be arbitrarily large (except in the degenerate case where vertical is one of the extreme directions of a bundle). When there are no vertical bundles, the largest vertical extent of the bundles is usually just the smallest vertical extent of the set of lines that support the boundaries of the bundles, which can be computed in linear time. We need to be careful that the defining lines of the result do not belong to the same bundle, but this can be ensured; details are in the full paper.

4.2 Smallest value

We want to compute the shortest vertical segment that stabs all of the bundles. Again, we separate the case where none of the bundles contain vertical lines, or when some of them do.

No vertical bundles. When there are no vertical bundles, we can view the lines as directed lines from left to right, and then the problem is very similar to linear programming on upfacing bundles. We show in the full paper that a similar approach also yields an $O(n \log n)$ algorithm for this problem.

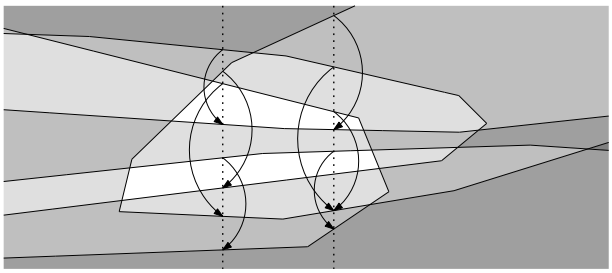


Figure 3: An arrangement with four bundles. One of the bundles does not contain a vertical line, the other three do. The dotted line is the current sweepline, and the arrows point to the highest possible lower endpoint of an interval with the given upper endpoint that stabs all bundles. Between the two lines, there has been one event.

General. If there are vertical bundles, then we can solve the problem in $O(n^2)$ time. The difference with the previous case is that there any vertical line would cross all bundles in exactly one interval. Now it may cross a bundle in an 'inverted' interval, that is, there is an interval where the bundle is *not* crossed but outside the interval it is.

When we are given the vertical line that contains the optimum, we need to solve a 1-dimensional problem. Then we want the shortest interval that intersects n 'inverted' intervals (or a mixture of normal and inverted intervals). We can do this in linear time (after sorting), because we can sweep an interval down the line and both ends move in only one direction.

To solve the problem in quadratic time, we will sweep a vertical line from left to right. The current line will intersect a number of bundle boundaries. We call such an intersection a *ceiling* if it has the interior of the bundle above it, and a *floor* if it has the interior of the bundle below it. We maintain the following structure: a sorted list with all bundle boundaries currently intersected by the sweepline, and for all ceilings we store the a pointer to the highest floor (below it) such that the segment between the ceiling and the floor intersects all bundles. We also need to maintain a list of pointers from any floor to all ceilings pointing to it. Figure 3 shows these pointers for some lines.

We have an event whenever two boundaries cross, or when a ceiling and a floor meet and vanish or appear. Most events can easily be taken care of; we will describe the most interesting one here.

When two ceilings intersect, the highest of the two (before the event) must now start caring about one extra bundle. If it already cared, nothing needs to change, otherwise its pointer must jump to the floor that corresponds to the same bundle that we crossed. The lowest of the two can now stop caring about one bundle. Only if it was currently pointing to floor that corresponds to that bundle, something needs to happen. In this case, its pointer must jump to the place the other ceiling was pointing to! This event takes constant time. Figure 3 shows the situations before and after a ceiling-ceiling event.

This completes the $O(n^2)$ algorithm; all events are described in detail in the full paper, where we also show how to generalise this algorithm to $O(n \log n + m^2)$ when only m of the bundles contain vertical lines. An interesting open question is whether a faster algorithm exists.

References

[1] P. Bhattacharya and A. Rosenfeld. "Convexity" of sets of lines. *Pat. Rec. Let.* 19:1199–1205, 1998.

[2] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*, *EATCS Monographs on Theoretical Computer Science* 10. Springer-Verlag, 1987.

[3] A. Efrat, G. Rote, and M. Sharir. On the union of fat wedges and separating a collection of segments by a line. *Comput. Geom. Theory Appl.*, 3:277–288, 1993.

[4] A. Gajentaan and M. H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom. Theory Appl.*, 5:165–185, 1995.

[5] J. Gates. Some dual problems of geometric probability in the plane. *Combin. Probab. Comput.*, 2:11–23, 1993.

[6] J. E. Goodman. When is a set of lines in space convex? *Not. Am. Math. Soc.*, 45:222–232, 1998.

[7] J. Hershberger. Finding the upper envelope of n line segments in $O(n \log n)$ time. *Inform. Process. Lett.*, 33:169–174, 1989.

[8] D. J. Huggard. A linear programming model of herbivore foraging: imprecise, yet successful? *Oecologia*, 100(4):470–474, 1994.

[9] D. T. Lee and Y. F. Wu. Geometric complexity of some location problems. *Algorithmica*, 1:193–211, 1986.

[10] J. Matoušek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. *Algorithmica*, 16:498–516, 1996.

[11] A. Rosenfeld. "Geometric properties" of sets of lines. *Pat. Rec. Let.*, 16:549–556, 1995.

[12] M. van Kreveld and M. Löffler. Largest bounding box, smallest diameter, and related problems on imprecise points. In *Proc. 10th Worksh. Algo. Data Struct.*, LNCS 4619, pages 447–458, 2007.

The Linear Parametric Geometric Uncertainty Model: Points, Lines and their Relative Positioning

Yonatan Myers and Leo Joskowicz *

Abstract

Characterizing geometric uncertainty is a central problem in mechanical CAD/CAM, robotics, and computer vision. Geometric uncertainty is often studied with simple ad-hoc models that assume independent, isocentric, and isotropic geometric errors. These models are often inadequate and can lead to erroneous conclusions. In this paper, we introduce the Linear Parametric Geometric Uncertainty Model, and derive the worst-case first-order approximation of the uncertainty zones of points and lines in the plane. The model is general and expressive, and allows parameter dependencies. We present the properties of point and line uncertainty zones, and algorithms to compute them and to answer relative positioning queries.

1 Introduction

Geometric uncertainty plays a central role in many fields, including mechanical CAD/CAM, robotics, and computer vision. While geometric models of physical objects and their relative locations are exact, in practice, manufacturing and measurement processes introduce geometric uncertainties.

Modern tolerancing and metrology has extensively studied the shape and position uncertainty of parts. Despite recent advances, their characterization and efficient computation remains an open problem [1].

Many models of geometric uncertainty have been proposed. One approach bounds point variations with simple geometric entities, such as rectangles, circles [2, 3], convex polygons [4], edge variations [6], and boundary offsets [5]. This assumes that point uncertainties are isocentric and independent, which often overestimates the real geometric uncertainty. A second approach models geometric feature variations with intervals of the coefficients of their algebraic parameterization [7, 8]. This accounts for features but not for the parameter dependencies between them. A third approach models variations with infinitesimal rigid body transformations that are concatenated to propagate the geometric uncertainty [9]. This allows for parameter dependencies but is limited to points and does not explicitly describe the uncertainty zone.

*School of Engineering and Computer Science, The Hebrew University of Jerusalem, ISRAEL yoni_m@cs.huji.ac.il, josko@cs.huji.ac.il

Two types of geometric uncertainty analysis methods are available. Sampling-based methods analyze the sensitivity to parametric variations by sampling parameter instances [9, 10]. This heuristic approach is general but computationally intensive. Geometry-based methods compute uncertainty zones with offsetting operations and bounding volumes [2, 3, 5, 6]. They derive zones which minimally contain all valid part instances. However, the resulting uncertainty part models are often too simplistic.

Computational geometry research focuses on the robustness and accuracy of basic geometric algorithms, such as line intersections and convex hulls. With few exceptions, they do not consider geometric uncertainty, nor describe algorithms for computing it.

In this paper, we introduce the Linear Parametric Geometric Uncertainty Model (LPGUM) and derive the worst-case first-order approximations of the uncertainty zones of points and lines in the plane. The model is based on our prior work [11, 12] and is general and expressive. We describe efficient algorithms to compute point and line uncertainty zones and to answer relative positioning queries.

2 The Linear Parametric Geometric Uncertainty Model (LPGUM)

In the LPGUM, geometric entities are defined by joint parameters, each with a nominal value and uncertainty interval. First-order linear dependency and geometric uncertainty are modeled by a sensitivity matrix. The *uncertainty zone* is the union of all instances of an entity, resulting from all parameter value combinations. The *uncertainty envelope* is the uncertainty zone's boundary.

Formally, a *parametric uncertainty model* $PM = (p, \bar{p}, \Delta)$ is defined by a vector p of k parameters over a domain Δ. The parameters' *uncertainty domain* is the cross product of the individual parameter domains Δ_i, each defined by an *uncertainty interval* $\Delta_i = \left[p_i^-, p_i^+\right]$, where $p_i^- < p_i^+$, and $p_i^-, p_i^+ \in \mathbb{R}$. Each parameter p_i has a *nominal value* $\bar{p}_i \in \Delta_i$, which is the parameter's value with no uncertainty. The *nominal parameters vector* $\bar{p} = (\bar{p}_1, ..., \bar{p}_k)$ is the vector of the parameter values with no uncertainty.

A *parametric point* $v(p) = (x(p), y(p), z(p))$ is a point defined by three continuous functions $x(p), y(p), z(p)$ over the uncertainty domain Δ of

parameter vector p. The *nominal point* $v(\bar{p}) = (x(\bar{p}), y(\bar{p}), z(\bar{p}))$, $\bar{p} \in \Delta$, is the location of the point with no uncertainty. The *uncertainty zone* of parametric point $v(p)$ is the set of all point instances: $\mathcal{V}(v) = \{v(p) \mid p \in \Delta\}$. The uncertainty zone is a closed, connected set, whose boundary is defined by the functions $x(p), y(p), z(p)$. The geometric interpretation of the uncertainty zone is that point v can be anywhere in the uncertainty zone, depending on the actual values of its parameters.

In general, the uncertainty zone boundary of a point cannot be derived analytically or computed exactly. However, since the parameters' uncertainty intervals are usually one order of magnitude smaller or less than the nominal value, the standard approach is to approximate them as linear deviations from the nominal value. The deviations are the partial derivatives of the point's functions evaluated at nominal parameter values:

$$v(p) \approx v(\bar{p}) + \sum_{i=1}^{k} \left(\frac{\partial v(\bar{p})}{\partial p_i} \right) \psi_i \qquad (1)$$

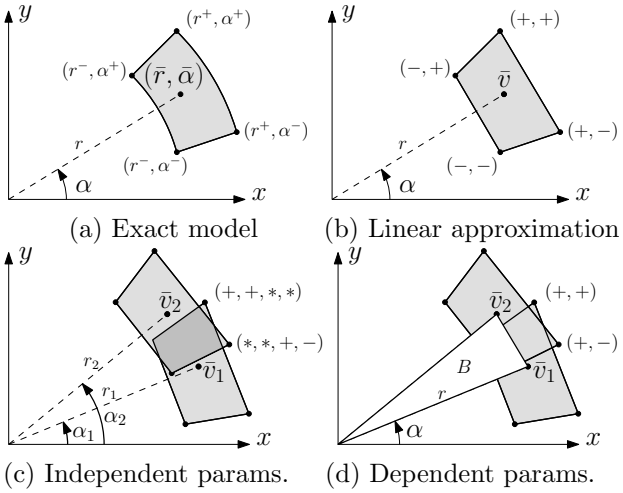
where $\psi_i = (p_i - \bar{p}_i)$ is the i^{th} parameter offset, and $\frac{\partial v(\bar{p})}{\partial p_i} = \frac{\partial v(p)}{\partial p_i}|_{p=\bar{p}}$ is the partial derivative of $v(p)$ with respect to p_i evaluated at $\bar{p}$. The constants $\frac{\partial v(\bar{p})}{\partial p_i}$ can be grouped in a $3 \times k$ *uncertainty sensitivity matrix*:

$$A_v = \left(\begin{array}{ccc} \frac{\partial x(p)}{\partial p_1} & \cdots & \frac{\partial x(p)}{\partial p_k} \\ \frac{\partial y(p)}{\partial p_1} & \cdots & \frac{\partial y(p)}{\partial p_k} \\ \frac{\partial z(p)}{\partial p_1} & \cdots & \frac{\partial z(p)}{\partial p_k} \end{array} \right)_{|\ p=\bar{p}} \qquad (2)$$

Rows represent the sensitivity of the point coordinates x, y, z to variations in the values of parameters p. Columns, $(A_v)_i$ represent the sensitivity of the point coordinates to variations in the parameter p_i value. A point's LPGUM, $LPGUM(v) = (v(\bar{p}), p, \bar{p}, \Delta, A_v)$ is defined by the nominal point location $v(\bar{p})$, a vector p of k parameters and their uncertainty domain Δ, the nominal values vector $\bar{p}$, and a $3 \times k$ uncertainty sensitivity matrix A_v. Following Eq. 1, the LPGUM of point v, is thus $v(p) = v(\bar{p}) + A_v(p - \bar{p})$.

The LPGUM allows for dependent parameters and asymmetric uncertainty intervals where $|p_i^- - \bar{p}| \neq |p_i^+ - \bar{p}|$. We transform an asymmetric model to a symmetric one by substituting $PM = (p, \bar{p}, \Delta)$ by $PM' = (p, \bar{p}', \Delta)$ where $\bar{p}'_i = \frac{p_i^+ + p_i^-}{2}$. To simplify the notation we denote $(p - \bar{p}')$ as q.

We illustrate the LPGUM with an example (Fig. 1). Let $v(p)$ be a point in the plane defined in radial coordinates by two parameters: r, the distance from the origin, and α the angle from the horizontal. Let $p = (r, \alpha)$ be the parameters vector and $\bar{p} = (\bar{\alpha}, \bar{r})$ the nominal parameters vector. Let the parameter uncertainty intervals be $[\alpha^-, \alpha^+]$ and $[r^-, r^+]$. The point uncertainty zone is a disc sector, with inner and outer radii, r^- and r^+, at angles α^- and α^+(Fig. 1a). The



(a) Exact model (b) Linear approximation

(c) Independent params. (d) Dependent params.

Figure 1: Uncertainty models of a point. (a) Exact model, and (b) linear approximation (the uncertainty zone is shaded). The sign vectors indicate the extreme parameter offsets that define the endpoints (− min, + max). Two-point models: (c) independent, and (d) dependent parameters. The common uncertainty zone is shaded dark.

four endpoints correspond to the extremal parameter values combinations. The sensitivity matrix A_v is:

$$A_v = \left(\begin{array}{cc} \frac{\partial(r\sin\alpha)}{\partial r} & \frac{\partial(r\sin\alpha)}{\partial \alpha} \\ \frac{\partial(r\cos\alpha)}{\partial r} & \frac{\partial(r\cos\alpha)}{\partial \alpha} \end{array} \right)_{|(r=\bar{r}, \alpha=\bar{\alpha})} = \left(\begin{array}{cc} \sin\bar{\alpha} & \bar{r}\cos\bar{\alpha} \\ \cos\bar{\alpha} & -\bar{r}\sin\bar{\alpha} \end{array} \right)$$

The approximate uncertainty zone is a rectangle defined by extremal parameter values (Fig. 1b).

To illustrate parameter dependencies, consider now two uncertain points and their relative position. When the point parameters are independent and the uncertainty zones overlap (Fig. 1c), their relative ordering can change. In the common zone, some instances of v_2 might be below point instances of v_1. Outside the common zone, the order is preserved. When the points depend on the same parameters (e.g., they are on rigid body B), their relative order is always preserved, although their uncertainty zones overlap (Fig. 1d).

3 LPGUM of a point

Consider the uncertainty envelope of a point in the plane defined in the LPGUM model.

$$v(q) = \bar{v} + A_v q \qquad (3)$$

Finding the extreme vertex of the uncertainty envelope in a given direction d is equivalent to finding the point in direction d that maximizes $\langle A_v^\top d, q \rangle$. The vertex is obtained by solving the linear program:

$$\max_q \langle A_v^\top d, q \rangle \quad \text{subject to:} \quad q \in \Delta$$

The maximization is done separately for every parameter q_i, as the parameter inequalities are independent.

To compute the point uncertainty envelope; sort the column vectors of the sensitivity matrix by their angle, build the cone diagram and compute a sign vector for one cone. Then iterate over all cones according to the angle of the lines bounding them. For every cone flip the parameter associated with the line crossed between p^+ and p^-, and compute the vertex.

Theorem 1 *Let $v(q) = \bar{v} + A_v q$ be an LPGUM point in the plane, dependent on k parameters. Its uncertainty envelope is a convex polygon with at most $2k$ vertices which can be computed in optimal $O(k \log k)$ time, $O(k)$ space. When the parameters q are defined over symmetric intervals, the uncertainty envelope is a centrally symmetric convex polygon (zonotope).*

4 LPGUM of a line

The LPGUM of a *parametric line* l, defined by a point $v(q)$ and a vector $u(q)$, with sensitivity matrices A_v and A_u, is the affine combination of the parametric point and the direction vector times a scalar, $\alpha \in \mathbb{R}$:

$$l(q) = l(v(q), u(q)) = \bar{v} + A_v q + \alpha(\bar{u} + A_u q) \quad (4)$$

Theorem 2 *Let $l(q)$ be an LPGUM line defined by Eq. 4 dependent on k parameters. If the line is bounded it has two boundaries each of which has $O(k)$ vertices. The regions outside the uncertainty zone are open and convex.*

To compute the uncertainty envelope of an LPGUM line $l(q)$, we sweep the values of α. Every value of α yields a zonotope, $l_\alpha(q)$. While α changes, $l_\alpha(q)$ traces out the line's envelope. The sweep stops at values of α called *events*, where the envelope changes. The algorithm iterates over all the events, tracing out the uncertainty envelope segments, and finding new events. When the event queue is empty, the segments are combined. The algorithm runs in $O\left(k^2 \log k\right)$ time and $O\left(k^2\right)$ space. We identify three types of events:

1. *Switch events* occur at values of α where two lines of the cone diagram of the point $l_\alpha(q)$, coincide. To find the events, we solve, $\left\langle (A_\alpha)_i, (A_\alpha)_j^\perp \right\rangle = 0$.
2. *Flip events,* occur at values of α for which a column vector $(A_\alpha)_i$ equals zero. To find the events, we solve $(A_\alpha)_i = 0$ for α, for every column.
3. *Twist events* occur at α values for which the part of $l_\alpha(q)$ in contact with the uncertainty envelope changes. To find the events, we solve $l\left(q^{(i)}\right) = l\left(q^{(j)}\right)$ for the two values of α, where $q^{(i)}$, $q^{(j)}$ are neighboring vertices parameterizations on the zonotope $l_\alpha(q)$.

To move the zonotope from event to event, we connect points on the two zonotope boundaries with the same parameterization by a line segment. The swept segments generate a quadratic curve.

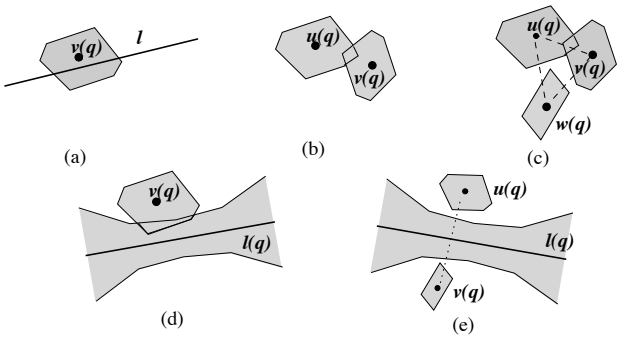


Figure 2: Examples of relative position relations between uncertain points and lines (uncertainty zones are shaded): (a) classification of an uncertain point with respect to a line; (b) relative position of two uncertain points; (c) relative position of three uncertain points; (d) classification of an uncertain point with respect to an uncertain line; (e) uncertain perpendicular bisector of two uncertain points.

5 Relative position relations

We study five types of relative position relations in the LPGUM. Figure 2 illustrates each case.

5.1 Uncertain point/nominal line classification

Classifying an uncertain point with respect to a nominal line is equivalent to classifying a convex polygon with respect to a line, which takes $O(n \log n)$ time.

When the nominal line intersects the point's zonotope, we compute the parameter values for which the point instances lie on the line or on either side of it. Let l be a line in closed form, $y = ax + b$ and let v be an LPGUM point. The classification of the point instances depends on the value of q and is determined by:

$$(\bar{v} + A_v q)_y < a(\bar{v} + A_v q)_x + b \quad (5)$$

$$a(\bar{v})_x - (\bar{v})_y + \left(a(A_v)_x - (A_v)_y\right) q + b > 0 \quad (6)$$

This k-dimensional hyperplane, which cuts the hypercube in parameter space, partitions the point instances into points above, on or below the line.

5.2 Relative position of two uncertain points

Given two points $u(q)$, $v(q)$, we define the vector $\vec{d} = \overrightarrow{\bar{u}\bar{v}}$ as the nominal direction from u to v. To see whether for a parameterization q the vector $\vec{d}(q) = \overrightarrow{u(q)\, v(q)}$ is in the opposite direction to $\overrightarrow{\bar{u}\bar{v}}$, we check to see whether $\left\langle \vec{d}, \vec{d}(q) \right\rangle < 0$. To determine whether a direction flip can occur, we see if:

$$\min_q \left\langle \bar{v} + A_v q - \left(u + \bar{A}_u q\right), \bar{v} - \bar{u} \right\rangle \quad \text{subj. to: } q \in \Delta$$

is negative.

5.3 Relative position of three uncertain points

To determine whether a point $w(q)$ in the plane lies to the left or right of the line from $u(q)$ to $v(q)$, we consider the determinant of the matrix

$$M(q) = \begin{bmatrix} u_x + (A_u)_x q & u_y + (A_u)_y q & 1 \\ v_x + (A_v)_x q & v_y + (A_v)_y q & 1 \\ w_x + (A_w)_x q & w_y + (A_w)_y q & 1 \end{bmatrix}$$

$\det(M(q))$ is a multivariate quadratic expression (with q_i as its variables) for which we must determine which of the following three cases holds.
1. $\exists q^{(1)}, q^{(2)} \in \Delta$ such that, $\det\left(M\left(q^{(1)}\right)\right) < 0$ and $\det\left(M\left(q^{(2)}\right)\right) > 0$, for some parametrizations $w(q)$ is to the left and for others it is to the right.
2. $\forall q \det(M(q)) > 0$, $m(q)$ is always to the left.
3. $\forall q \det(M(q)) < 0$, $m(q)$ is always to the right.

To decide the second case, which is the exact opposite of the third, we solve Eq. (7). If the constraint extremum on the determinant is > 0 the case holds.

$$\min_q \det(M(q)) \quad \text{Subject to:} \quad q \in \Delta \tag{7}$$

5.4 Uncertain point/uncertain line classification

To classify an LPGUM point $v(q)$ with respect to an LPGUM line, $l(q)$, we define two points on the line $u(q) = l_{\alpha_1}(q)$ and $w(q) = l_{\alpha_2}(q)$, $\alpha_1 < \alpha_2$ and solve the three point classification question above. If case (1) holds the point is on different sides of the line for different qs. If case (2) holds the point is to the left of the line; otherwise it is to the right.

5.5 Perpendicular bisector of two uncertain points

The perpendicular bisector of a segment connecting two uncertain points is defined as follows. Two LPGUM lines $l(q)$ and $m(q)$ are *perpendicular* to each other if for every parameterization t, $l(t) \perp m(t)$.

Let $u(q)$ and $v(q)$ be LPGUM points. An LPGUM point $a(q)$ is said to *divide the segment* $\overline{u(q)\,v(q)}$ *in the proportion* $\lambda \in [0,1]$, if for any parameterization t, $a(t) = \lambda u(t) + (1-\lambda) v(t)$. For $\lambda = \frac{1}{2}$, the point $a(q)$ is the mid-point of the segment $\overline{u(q)\,v(q)}$.

To compute the perpendicular bisector, we find the point at the center of the nominal segment:

$$v_{mid}(q) = \frac{\bar{u} + \bar{v}}{2} + \left(\frac{A_u + A_v}{2}\right) q \tag{8}$$

To find the perpendicular to the line from u to v, we compute the vector $\vec{d}(q) = v(q) - u(q)$ and rotate it by $\frac{\pi}{2}$ by rotating the nominal part and all column vectors of the sensitivity matrix. All rotations must be in the same direction. The perpendicular bisector to $\overline{v(q)\,u(q)}$ is:

$$l^{\perp}(q) = v_{mid}(q) + \alpha \vec{d}^{\perp}(q) \tag{9}$$

6 Conclusion

This paper presents a parametric, first-order model for the geometric uncertainty of points and lines in the plane. The model is general, expressive, accounts for parameter dependencies and for asymmetric zones. We present the LPGUM of a point and a line in the plane and describe properties and algorithms to efficiently compute their uncertainty zones and to determine their relative position relations. In the future, we plan to explore the properties of uncertain convex hulls, Voronoi diagrams, and line arrangements, and develop efficient algorithms for their computation.

References

[1] ASME Y14.5 Dimensioning and Tolerancing Standard, and ASME Y14.5M Mathematical Definition of Dimensioning and Tolerancing Principles. *American Soc. Mech. Engs.*,, New York, 1994.

[2] S. Akella and M. Mason. Orienting toleranced polygonal parts. *International Journal of Robotics Research*, 19(12):1147–1170, December 2000.

[3] J. Chen, K. Goldberg, M. Overmars, D. Halperin, K.-F. Böhringer, and Y. Zhuang. Computing tolerance parameters for fixturing and feeding. *The Assembly Automation Journal*, 22:163–172, 2002.

[4] R.C. Brost and R.R. Peters. Automatic design of 3D fixtures and assembly pallets. In *IEEE International Conference on Robotics and Automation*, pages 495–502, Minneapolis Minnesota, April 1996. IEEE Int.

[5] J.R. Rossignac and A. A. G. Requicha. Offsetting operations in solid modelling. *Computer Aided Geometric Design*, 3(2):129–148, 1986.

[6] J.-C. Latombe, R.H. Wilson, and F. Cazals. Assembly sequencing with toleranced parts. *Computer-Aided Design*, 29(2):159–174, 1997.

[7] S. Gupta and J.U. Turner. Variational solid modeling for tolerance analysis. *IEEE Comput. Graph. Appl.*, 13(3):64–74, 1993.

[8] U. Roy and B. Li. Representation and interpretation of geometric tolerances for polyhedral objects. ii.: Size, orientation and position tolerances. *Computer-Aided Design*, 31(4):273–285, 1999.

[9] D.E. Whitney. *Mechanical Assemblies: Their Design, Manufacture, and Role in Product Development*. Oxford University Press, USA, February 2004.

[10] M. Giordano, E. Pairel, and S. Samper. Mathematical representation of tolerance zones. In F. van Houter and H. Kale, editors, *Global Consistency of Tolerances*, pages 177–186, Kluwer Academic Publishers, London, March 1999.

[11] Y. Ostrovsky-Berman and L. Joskowicz. Tolerance envelopes of planar mechanical parts with parametric tolerances. *Computer-Aided Design*, 37(5), 2005.

[12] Y. Ostrovsky-Berman and L. Joskowicz. Relative position computation for assembly planning with planar toleranced parts. *The International Journal of Robotics Research*, 25(2):147–170, 2006.

Smoothing Imprecise 1-Dimensional Terrains *

Chris Gray† Maarten Löffler‡ Rodrigo I. Silveira‡

Abstract

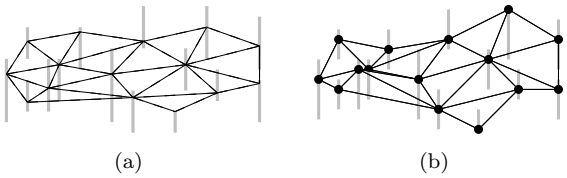
An imprecise 1-dimensional terrain is an x-monotone polyline where the y-coordinate of each vertex is not fixed but only constrained to a given interval. In this paper we study four different optimization measures for imprecise 1-dimensional terrains, related to obtaining smooth terrains. In particular, we present algorithms to minimize the largest and total turning angle, and to maximize the smallest and total turning angle.

1 Introduction

Terrain modeling is a central task in geographical information systems (GIS). Terrain models can be used in many ways, for example for visualization or analysis purposes (to compute features like watersheds or visibility regions [1]). One common way to represent a terrain is by means of a triangulated irregular network (TIN): a planar triangulation with additional height information on the vertices.

This height information is often collected by airplanes flying over the terrain and sampling the distance to the ground, for example using radar or laser altimetry techniques, or it is sometimes obtained by optically scanning contour maps and then fitting an approximating surface. These methods often return a height interval rather than a fixed value, or produce heights with some known error bound. For example, in high-resolution terrains distributed by the United States Geological Survey, it is not unusual to have vertical errors of up to 15 meters [7]. However, algorithms in computational geometry often assume that the height values are precise. This may lead to artifacts in the terrain.

An alternative to deal with this imprecision in terrains is to use a more involved model that takes the imprecision into account. Gray and Evans [2] propose a model where an interval of possible heights is associated with every vertex of the triangulation. Figure 1 shows an example of an imprecise terrain, and a possible instance of the real terrain. Kholondyrev and Evans [5] also study this model. Silveira and Van Oostrum [6] also allow moving vertices of a TIN up and down to remove local minima, but do not assume bounded intervals.



(a) (b)

Figure 1: (a) An imprecise terrain. (b) A possible instance of the real terrain.

In this paper we use the same model as in [2, 5]; each vertex has a height interval. This leads to some freedom in the terrain: the real terrain is unknown. This paper deals with trying to obtain a *smooth* terrain, respecting the height intervals. This may be needed either because some additional information about the morphology of the terrain is known (that is, there are not too many sharp ridges in that area) or for visualization or compression purposes. Smoothing of terrains has been previously studied in the context of grid terrains [4, 7], where techniques from image processing can be applied, but not, to our knowledge, for imprecise TINs. In a TIN, a smooth terrain implies that the spatial angles between triangle normals are not too large. We can try to find a height value for each vertex, restricted by the intervals, such that the resulting terrain minimizes the largest spatial angle, or the sum of all spatial angles.

We study these measures, and two other ones, but only for 1-dimensional terrains. A 1-dimensional terrain is essentially an x-monotone polyline. An imprecise 1-dimensional terrain is the same thing, but with a y-interval for each vertex rather than a fixed coordinate—see Figure 2. This work constitutes a first step towards solving the 2-dimensional case.

We study four variants of the problem. In Section 2 we minimize the sum of the turning angles of the polyline, while in Section 3 we minimize the largest one. Both measures aim to smooth the terrain as much as possible; which is best to use depends on the situation at hand. In Section 4 we maximize the sum of the turning angles, and in Section 5 we maximize the smallest one. These measures aim to make the terrain as rough as possible: this gives some idea of the worst possible case.

*This research was partially supported by the Netherlands Organisation for Scientific Research (NWO) through the project GOGO and project no. 639.023.301.

†Department of Computing Science, TU Eindhoven, the Netherlands, cgray@win.tue.nl

‡Department of Information & Computing Sciences, Utrecht University, the Netherlands, {loffler,rodrigo}@cs.uu.nl

In the discussion of the algorithms below, we assume that the 1-dimensional terrain is an x-monotone polyline. In this context, we define a *left-turn* and *right-turn* as a vertex where this polyline, seen from left to right, turns to the left (upwards) or right (downwards) respectively.

2 Minimizing the total turning angle

To minimize the sum of (the absolute values of) the turning angles, we make the following observations. When we leave a certain vertex in a certain direction, and we enter another vertex from a certain direction, and the path between them is left-turning (or right-turning), then it does not matter how exactly this path goes: the total turning angle stays the same. This means there will most likely be many equivalent optimal solutions. An example is shown in Figure 2(a).

In fact, if the leftmost and rightmost intervals would just be single points, the shortest path between those points through the corridor between the polylines defined by the upper and lower endpoints of the intervals is one of the optimal solutions. To see why, consider the global shape of the shortest path. This will be a polyline, with a number of left and right turns. Each right turn must lie on the lower endpoint of its imprecision interval, otherwise there would be a shorter path possible. Similarly, each left turn must lie on the upper endpoint of its interval. Now call a segment of the shortest path *critical* when it has one left and one right turn. It is not hard to see that we cannot get a better turning angle than the sum of the turning angles between pairs of consecutive critical segments. However, the total turning angle of the shortest path achieves exactly this lower bound, and hence is optimal.

When the leftmost and rightmost intervals are not single points but real intervals, we need to make one additional observation. If we compute the shortest path from some point on the leftmost interval to some point on the rightmost interval, then this may contain some unnecessary turns at the ends. We can identify the leftmost and rightmost critical segments of the path, and note that the best thing to do is just continue in a straight line from these segments towards the leftmost and rightmost intervals since then we make no extra turns. It can be that this is not possible, but in that case we just make the turn at those extreme critical segments as small as possible.

The shortest path can be computed in linear time [3]. The adaptations that are necessary at the ends are also easy to do in linear time.

3 Minimizing the largest turning angle

It appears that the problem of minimizing the maximum angle in a realization of an uncertain terrain is difficult. The main problem is that finding a solution requires finding the inverse of a non-algebraic function. Therefore, we present an approximate solution. If the best realization of a given uncertain terrain has a maximum turning angle α, we find a terrain with maximum turning angle at most $\alpha + \varepsilon$, for any given ε. Our solution is a dynamic-programming algorithm that works by discretizing the range of angles that we consider.

Figure 2(b) shows an example of a path which minimizes the largest turning angle.

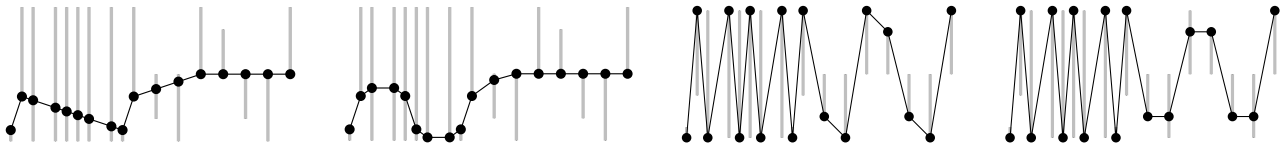
Let D be a set of $k = \lceil \pi/\varepsilon \rceil$ angles, evenly spaced from $-\pi/2$ to $\pi/2$. Our algorithm is based on solving subproblems of the form $S[p_1, p_2, d_1, d_2]$, where p_1 and p_2 are endpoints of intervals and d_1 and d_2 are directions from D. It is also based on the following observation: let P be the optimal path through a set of intervals between points p_1 and p_2 that are assumed to have infinite length. If we remove the assumption that the intervals are of infinite length and P no longer passes through all of the intervals, then the optimal path must pass through the endpoint of at least one of the intervals in the set. This allows us to find the optimal path recursively by trying all endpoints of all intervals between p_1 and p_2 and all directions from D.

We begin by describing the process of finding the optimal path through a set of intervals, assuming that the lengths of all the intervals are infinite. Let P^* be the optimal path between p_1 and p_2 that leaves p_1 at angle d_1 with respect to horizontal and enters p_2 at angle d_2 with respect to horizontal. It is easy to see that at almost all of the vertices of P^*, the turning angle is the same (call it θ^*). There may be one vertex at which the turning angle is smaller than θ^*, if the direction of curvature changes, but this happens at most once. We can find an ε-approximation to P^* by binary searching for θ^*.

We perform the binary search by constructing a path P_1 starting at p_1 and a path P_2 from right to left starting at p_2. We construct P_1 and P_2 so that all the turning angles are some common θ. The angle between P_1 and P_2 at their intersection allows us to determine whether we should raise or lower θ. When the angle at the intersection of P_1 and P_2 becomes less than ε, we stop the binary search. Since the range of angles for θ is bounded above by $\pi/2$, the time complexity for the binary search is $O(n \log 1/\varepsilon)$.

Note that the direction of the turns in P_1 and P_2 is not specified. In fact, there are four options depending on whether P_1 or P_2 are left-turning or right-turning. We perform the binary search for each of these options, keeping the search which returns the lowest θ.

As in Section 2, we initially reduce the first and last intervals to one of their endpoints. We call these points p_1 and p_n. We then find the



(a) (b) (c) (d)

Figure 2: Four different measures applied to the same imprecise terrain. Optimal terrains for (a) min total turning angle, (b) min max angle, (c) max total turning angle and (d) max min angle.

lowest value of $S[p_1, p_n, d_1, d_2]$ over all values of d_1 and d_2. We construct the table S by combining the results of recursively-computed subproblems. We set $S[p_i, p_k, d_i, d_k] = \min_{p_j, d_j, d'_j} \max\{|d_j - d'_j|, S[p_i, p_j, d_i, d_j], S[p_j, p_k, d'_j, d_k]\}$.

Constructing the table S takes $O(n^2k^2)$ space and $O(nk^2 + n \log k)$ per entry. Therefore the entire algorithm takes $O(n^3(1/\varepsilon)^4)$ time.

4 Maximizing the total turning angle

When analyzing imprecise terrains, we may also be interested in the worst possible case for the real terrain. When our goal is to maximize the sum of the turning angles, we make the simple observation that we only need to consider the endpoints of the imprecision intervals. Indeed, if we have a solution that uses some point on the interior of an interval, there is at least one direction in which we can move the point such that the total turning angle does not decrease: if it lies on a left-turning or right-turning chain, moving it will not change the total angle at all, and if it does not, then it is always good to make the turn sharper, since this increases both its own turning angle and that of (one of) its neighbors. See Figure 2(c) for an example.

After making this observation, it is not hard to come up with a linear time algorithm to solve the problem. We simply apply 1-dimensional dynamic programming: for each segment, defined by the upper/lower endpoints of two consecutive intervals, we store the optimal solution to the left of that segment that uses it. There are $4n$ such segments, and computing a value involves matching it with the two possible stored solutions to the left of it and picking the best.

5 Maximizing the smallest turning angle

We can also try to maximize the smallest turning angle to try to make the terrain as rough as possible. Figure 2(d) shows an example.

For this problem we present a linear time approximation algorithm. The terrain computed by the algorithm will have a minimum angle at least $(\alpha^*_{min} - \varepsilon)$, where α^*_{min} is the minimum angle in the optimal terrain and ε is any constant. It is based on going through the intervals from left to right, and computing the solutions of partial subproblems defined between the first (leftmost) interval and the current one.

Given an instance of an imprecise terrain, we call an interval *extremal* if its vertex is positioned at one of the endpoints of the interval. An non-extremal interval is called *internal*. With some abuse of notation, we will also refer to extremal/internal *vertices*.

We base the algorithm on the following observations:

- Let T^* denote the optimal terrain, with minimum angle α^*_{min}, and let α^*_i denote the angle of the vertex x_i of I_i in T^*. We assume that $\alpha^*_{min} > \varepsilon$, otherwise any terrain will be within the approximation factor.
- Let I_i be an internal interval. Then $\alpha^*_i = \alpha^*_{min}$. Moreover, I_i is part of a left-turning/right-turning chain of angle α^*_{min}.
- The leftmost and rightmost intervals of T^* are extremal.
- If there is an interval I_i in T^* connecting a left-turning chain to a right-turning chain which is immediately followed by another left-turning chain (or *vice versa*), then I_i is extremal. If it is not, we can take the middle right-turning chain as a whole and move it up, until I_i becomes extremal, and balance all the angles again, increasing the overall minimum angle.
- The maximum number of vertices of any left-turning or right-turning chain in T^* is $K = \lceil \pi/\varepsilon \rceil$. Therefore the maximum number of consecutive intervals that are internal is $2K$.

The previous observations imply that if a subproblem has more than $2K$ intervals, there must be at least one extremal interval among them. Extremal intervals allow us to separate the problem into independent subproblems. As in the algorithm of Section 3, we will consider only $k = \lceil 2\pi/\varepsilon \rceil$ possible directions.

Assume for now the following subproblem can be solved in constant time. $OptimalChain(p_i, p_j, d_i, d_j)$, for $i < j, (j - i) < 2K$, returns the solution to the

subproblem from I_i to I_j, with fixed positions p_i and p_j, and fixed incoming directions at I_i and I_j, given by d_i and d_j, under the assumption that all the intervals between I_i and I_j are internal. Notice that since there are at most $2K$ intervals, the whole subproblem has constant size.

5.1 Main algorithm

The algorithm goes through the intervals from left to right. Solutions to partial subproblems are stored in a table with entries of the shape $S[j, p, d]$. Such an entry stores the value (and information to reconstruct the terrain) of a solution for the terrain going from I_1 to I_j, finishing at I_j at position p (top/bottom extreme) with direction d. We explain how to compute the value of $S[j, p, d]$, assuming that the values for $S[i, p, d]$, $i < j$, for all possible positions p and directions d, have been already computed.

Assume that we are computing $S[j, p, d]$ for p one of the two extremes of I_j, and some incoming direction d (incoming at I_j). In order to find the value of $S[j, p, d]$ we need to know which is the first extremal interval found when going from I_j to I_1. The previous observations show that such an interval lies between $I_{(j-2K)}$ and I_{j-1}. We will consider each of them. For each interval choice I_r, we will also consider both choices for the position at I_r, p_r, and all k choices for the incoming direction d_r. This gives rise to a total of $2K \cdot 2 \cdot k$ combinations. For each of them we must solve the subproblem between I_r and I_j. To solve it we apply the algorithm $OptimalChain(p_r, p_j, d_r, d)$. The solution of $OptimalChain(p_r, p_j, d_r, d)$ is combined with the entry $S[r, p_r, d_r]$ to obtain the total value of this alternative. The best value among all the ones considered is stored at $S[j, p, d]$.

5.2 Solving the subproblems

The previous observations imply that any optimal solution for a series of intervals that does not use any extremal point must be a left-turning chain followed by a right-turning chain, or *vice versa*, with all vertices having turning angle α^*_{min}. It can also be comprised of only one left-turning or right-turning chain, but we see it as a degenerate case of the previous ones, hence we consider two possible shapes: left-turning/right-turning and right-turning/left-turning.

To find an approximation of the optimal chain, we follow an approach similar to the one used in Section 3. We first guess the minimum angle α_{min} and the shape of the chains. Once the general shape of the chain is known, we construct a chain that at every vertex turns exactly α_{min}. If the final chain reaches the last interval of the subproblem and it does it with an angle of at least α_{min}, we are done. Otherwise we need to try another value of α_{min}. Recall that the number of possible angles to try is only $\lceil \pi/\varepsilon \rceil$, and that the size of the subproblem is constant. Therefore the subproblem can be solved in constant time.

It is easy to verify that the angle of the solution computed by the algorithm is at most ε away from α^*_{min}. The running time is $O(n \cdot (1/\varepsilon)^4 \log(1/\varepsilon))$.

6 Conclusions

We studied several measures to compute the smoothest or least smooth possible 1-dimensional terrain, when height information is imprecise. We gave efficient algorithms for three cases, and a somewhat less efficient algorithm for the fourth case. Two of the algorithms are approximate.

Future work includes trying to improve the $O(n^3)$ algorithm for minimizing the maximum angle. We would also like to tackle 2-dimensional terrains. This poses challenges both at the modeling level (a definition of a *smooth* TIN is not straightforward) and also at the algorithm level.

References

[1] L. de Floriani, P. Magillo, and E. Puppo. Applications of computational geometry in Geographic Information Systems. In J. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 333–388. Elsevier, Amsterdam, 1997.

[2] C. Gray and W. Evans. Optimistic shortest paths on uncertain terrains. In *Proc. 16th Canadian Conference on Computational Geometry*, pages 68–71, 2004.

[3] L. J. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2:209–233, 1987.

[4] M. Hofer, G. Sapiro, and J. Wallner. Fair polyline networks for constrained smoothing of digital terrain elevation data. *IEEE Trans. Geosc. Remote Sensing*, 44:2983–2990, 2006.

[5] Y. Kholondyrev and W. Evans. Optimistic and pessimistic shortest paths on uncertain terrains. In *Proc. 19th Canadian Conference on Computational Geometry*, pages 197–200, 2007.

[6] R. I. Silveira and R. van Oostrum. Flooding countries and destroying dams. In *Proc. 10th Workshop on Algorithms and Data Structures*, LNCS 4619, pages 227–238, 2007.

[7] T. Tasdizen and R. T. Whitaker. Feature preserving variational smoothing of terrain data. In *Proceedings of the 2nd International IEEE Workshop on Variational, Geometric and Level Set Methods in Computer Vision*, 2003.

Noisy Bottleneck Colored Point Set Matching in 3D

Yago Diez * J. Antoni Sellarès *

Abstract

In this paper we tackle the problem of matching two colored point sets in $\mathbb{R}^3$ under the bottleneck distance. First we present an exact matching algorithm that requires the computation of intersections of complicated algebraic surfaces. To avoid this, we also present an approximate algorithm that is implementable and has improved asymptotic cost at the price of having the risk of "missing" some solutions. For the case of sets with very different cardinality, we speed up calculations by using a Lossless Filtering preprocess that discards the zones of the bigger set where matches cannot occur. We provide formal and experimental discussion on the running time and accuracy of the approximate algorithm (with and without Lossless Filtering preprocess).

1 Introduction

Protein molecules possess unique three-dimensional structures, defined by their amino-acid sequence, that has been found to determine many of their functional properties. Typically a protein molecule is modelled as a set of balls in $\mathbb{R}^3$, each representing an atom. Given a small collection of atoms, representing a secondary structure subunit or any other significant part of a protein (as some significant union of such secondary structures, called *motifs*), the *protein substructure detection problem* consists in determining whether the substructure exists in a protein molecule. We can see this problem as a colored point set matching problem. Since atom positions are fuzzy due to the finite precision of measuring devices, it is impractical to consider an exact match between two atoms. Bearing in mind that the correspondences between colored points to be one-to-one, we will use the *bottleneck distance*.

2 Problem formulation

Let $P(q,r)$ represent the colored point $q \in \mathbb{R}^3$ with associated color r. Given a real number $\epsilon \geq 0$, we say that two colored points $A = P(a,r)$,and $B = P(b,s)$ *match* when $r = s$ and $\tilde{d}(A,B) = d(a,b) \leq \epsilon$, where d denotes the Euclidean distance.

*Email: {ydiez,sellares}@ima.udg.es. IIA, Univ. de Girona, Spain. Partially supported by the Spanish Ministerio de Educación y Ciencia under grant TIN2007-67982-C02-02.

Let $\mathcal{D}$, $\mathcal{S}$ be two colored points sets of the same cardinality. A *color preserving bijective mapping* $f : \mathcal{D} \to \mathcal{S}$ maps each colored point $A = P(a,r) \in \mathcal{D}$ to a distinct and unique colored point $f(A) = P(b,s) \in \mathcal{S}$ so that $r = s$. Let $\mathcal{F}$ be the set of all color preserving bijective mappings between $\mathcal{D}$ and $\mathcal{S}$. The *bottleneck distance* between $\mathcal{D}$ and $\mathcal{S}$ is defined as:

$$d_b(\mathcal{D},\mathcal{S}) = \min_{f\in\mathcal{F}} \max_{A\in\mathcal{D}} \tilde{d}(A, f(A)) .$$

The **Noisy Colored Point Set Matching (NCPSM)** problem can be formulated as follows. Given two Colored Points sets $\mathcal{A}$, $\mathcal{B}$, $|\mathcal{A}| = n$, $|\mathcal{B}| = m$, $n \leq m$, and $\epsilon \geq 0$, determine all rigid motions τ for which there exists a subset $\mathcal{B}'$ of $\mathcal{B}$ such that $d_b(\tau(\mathcal{A}), \mathcal{B}') \leq \epsilon$. We define $\tau(P(a,r))$ as $P(\tau(a),r)$ and $\tau(A)$ as $\{\tau(P(a,r))|P(a,r) \in \mathcal{A}\}$

For the sake of clarity, through the rest of the paper we will just mention the colors associated to points only when necessary, thus we will speak about $a \in \mathcal{A}$ instead of $P(a,r) \in \mathcal{A}$.

2.1 Previous results

The study of the **NPSM** problem in $\mathbb{R}^2$was initiated by Alt *et al.* [2] who presented an exact $O(n^8)$ time algorithm for solving the problem for two sets $\mathcal{A}, \mathcal{B}$ of cardinality n. This cost can be reduced to $O(n^7 \log n)$ using the techniques in [6]. The 3D version of the problem is much less explored than its two dimensional counterpart. The only algorithms to the best of our knowledge are: an algorithm by Ambühl, Chakraborty and Gartner presented in [1] that suffered from a computational cost of $O(n^{32.5})$ and a $O(n^{13+5/6+\epsilon})$ result presented in [4].

3 NCPSM problem solving algorithm

We present an algorithm to solve the **NCPSM** problem that extends the 2D algorithm presented in [5], based in [6]. Due to space limitations, we focus in the aspects that are directly relevant to our problem, only sketch briefly the previous algorithm and present the main results omitting their proofs.

The algorithm consists on two phases: enumeration and testing. The enumeration phase makes the problem finite by partitioning all possible rigid motions of $\mathcal{A}$ into equivalence classes and choosing a representative motion τ for each class. The testing phase runs a matching algorithm between every set $\tau(\mathcal{A})$ and set $\mathcal{B}$.

As the enumeration phase requires the computation of an arrangement of complex surfaces, we propose an alternative, approximate version that is implementable and will overcome most of the problem-inherent complexity at the cost of "missing" some solutions. We will also dispose of a tradeoff between the computational time spent and the possibility of missing solutions.

3.1 Enumeration

Generating every possible rigid motion that brings set $\mathcal{A}$ onto a subset of $\mathcal{B}$ is infeasible due to the continuous nature of movement. We partition the set of all rigid motions in equivalence classes in order to make their handling possible. Consider the arrangement of spheres $\mathcal{S}_{\mathcal{B}} = \{S(b,\epsilon)|b \in B\}$. Any point $p \in \mathbb{R}^3$ belongs to the cell of the arrangement identified with the set of points $b \in \mathcal{B}$ that hold $p \in B(b,\epsilon)$.

Definition 3.1 *Two motions τ, μ are considered equivalent if and only if, for any colored point $a \in \mathcal{A}$ $\tau(a)$ and $\mu(a)$ lie in the same cell of $\mathcal{S}_{\mathcal{B}}$.*

Lemma 1 *Any rigid motion μ that is a solution to the* **NCPSM** *problem can be transformed to another motion μ' such that there exist three points $a_{i_1}, a_{i_2}, a_{i_3} \in \mathcal{A}$ holding that their images $\mu'(a_{i_1}), \mu'(a_{i_2}), \mu'(a_{i_3})$ lie on the spheres $S(b_{j_1},\epsilon), S(b_{j_2},\epsilon), S(b_{j_3},\epsilon)$ respectively where $b_{j_1} = \mu(a_{i_1}), b_{j_2} = \mu(a_{i_2})$ and $b_{j_3} = \mu(a_{i_3})$.*

As a consequence of Lemma 1 we can generate a representative for each equivalence class by considering each possible 6-tuple formed by three points in $\mathcal{A}$ and their counterparts in $\mathcal{B}$. As the matching must preserve colors, we also demand the colors associated to points to be matched to be the same.

Lemma 2 *Any given position of point $\tau(a_{i_1})$ in $S(b_{j_1},\epsilon)$ leaves a total of one degree of freedom for the positions of $\tau(a_{i_2})$ and $\tau(a_{i_3})$ in $S(b_{j_2},\epsilon)$ and $S(b_{j_3},\epsilon)$ respectively.*

Proof. For each possible position of point $\tau(a_{i_1})$, consider the geometric locus of all the points that belong to $S(b_{j_2},\epsilon)$ and whose distance to $\tau(a_{i_1})$ is exactly $d(a_{i_1}, a_{i_2})$. This corresponds, in the general case to a circle $C_{a_{i_1},a_{i_2}} \subset \mathbb{R}^3$ resulting from the intersection of $S(b_{j_2},\epsilon)$ and $S(\tau(a_{i_1}), d(a_{i_1}, a_{i_2}))$. Consider also the geometric locus of all the points that belong to $S(b_{j_3},\epsilon)$ and whose distance to $\tau(a_{i_1})$ is exactly $d(a_{i_1}, a_{i_3})$. Finally by choosing a point in one of the circles (determining, thus $\tau(a_{i_2})$) and imposing that $d(\tau(a_{i_2}), \tau(a_{i_3})) = d(a_{i_2}, a_{i_3})$ remains only a degree of freedom. □

The configuration space of our problem can be seen as a cube of side 2π where each dimension corresponds to one of the three angles that determine the associated rigid motion i.e. the two polar coordinates ϕ, ψ that determine the position of $\tau(a_{i_1})$ and the remaining angle θ corresponding to the point chosen in $C_{a_{i_1},a_{i_2}}$. From now on, we will denote $\tau_{\phi\psi\theta}$ the rigid motion that corresponds to any given value of parameters ϕ, ψ and θ. Another key observation is that, for any given values of parameters ϕ, ψ, θ any couple of the remaining points $a_{i_4} \in \mathcal{A}$, $b_{i_4} \in \mathcal{B}$ defines three possible positions corresponding to the position of $\tau_{\phi\psi\theta}(a_{i_4})$ with respect to the sphere $S(b_{i_4},\epsilon)$ (in, out or on the surface). These three possible positions correspond to the values of ϕ, ψ and θ for which a_{i_4} and b_{i_4} may (or may not) be matched. Consequently, given a 6-tuple $a_{i_1}, a_{i_2}, a_{i_3}, b_{j_1}, b_{j_2}, b_{j_3}$ and an additional couple a_{i_h}, b_{i_l} , we have a finite number of regions of $[0, 2\pi[^3$ that encode the information concerning when $\tau(a_{i_h})$ may be matched to b_{i_l} for a motion τ that brings $a_{i_1}, a_{i_2}, a_{i_3}$ to the boundary of the spheres $S(b_{j_1},\epsilon), S(b_{j_2},\epsilon), S(b_{j_3},\epsilon)$ respectively.

Consequently, in searching for the possible matchings, it suffices to consider the collection of $(n-3)(m-3) \in O(nm)$ surfaces of $[0, 2\pi[^3$ determined by the angles ϕ, ψ, θ such that $\tau_{\phi\psi\theta}(a_{i_h})$ belongs to the boundary of the sphere $S(b_{j_l},\epsilon)$ $a_{i_h} \notin \{a_{i_1}, a_{i_2}, a_{i_3}\}$ and $b_{j_l} \notin \{b_{j_1}, b_{j_2}, b_{j_3}\}$. Since we only need to encode the adjacency relationship among 3-dimensional cells of the arrangement we compute the vertical decomposition of the arrangement and then we compute the adjacency relationship of cells in the decomposition [7]. The number of cells of the vertical decomposition of an arrangement of n surfaces in $\mathbb{R}^3$ is $O(n^2\lambda_q(n))$, where q is a constant depending on the maximum degree of the surfaces, and $\lambda_s(k))$ is the maximum length of (k, s) Davenport-Schinzel sequence that is roughly linear in k [8]. The vertical decomposition of the arrangement can be computed in randomized expected time $O(n^{3+\epsilon})$, using the random-sampling technique [3]. Putting it all together, what we need to determine in order to generate a representative in every equivalent class of motions is the arrangement of surfaces in the cube $[0, 2\pi[^3$ defined by all possible couples once fixed a 6-tuple and then iterate over the set of all 6-tuples. This takes $O(n^{6+\epsilon}m^{6+\epsilon})$ expected time and $O(n^5m^5\lambda_q(nm))$ space.

3.2 Testing

Once we have generated all possible motions for set $\mathcal{A}$ in the Enumeration step we need to test if any of them matches some subset of set $\mathcal{B}$. To achieve this we use the bipartite matching algorithm presented in [5] for the two-dimensional case. In this algorithm we implicitly work with a bipartite graph that encodes adjacency relationships, to make them explicit

we use a compressed Octree. We generate values for parameters ϕ, ψ and θ inside each of the regions of the cube $[0, 2\pi[^3$ defined in the Enumeration section and test sets $\tau_{\phi\psi\theta}(\mathcal{A})$ and $\mathcal{B}$ for matching. By traversing the arrangement in order and maintaining the matching information for previous cells, the total amortized cost for testing each of the $O(n^5m^5\lambda_q(nm))$ cells is $O(n \log m)$. Consequently:

Lemma 3 *The computational cost of the exact matching algorithm is* $O(n^6m^5\lambda_q(nm) \log m)$.

As $\lambda_q(nm)$ is in $O(nm)$, this result meets the best theoretical results up to date [4].

4 Approximate algorithm

The complexity of the arrangement renders the algorithm difficult to implement and suffering from a very high computational cost. If we partition the search space using a cubic grid of side a given value γ, we achieve a $O(\frac{2\pi}{\gamma}^3 n^4m^4\lambda_q(nm))$ cost. This new algorithm is implementable although it is only an approximate algorithm. This has the problem that some of the cells of the arrangement may not be sampled and, consequently, some matches may be lost. More specifically, the matches that may be missed are those that have an associated cell in the arrangement that does not contain an axis-parallel cube of volume smaller than γ^3. We denote γ^3 as the *Slackness of the matching*. If we picture the process of finding matches as continuously moving a rigid copy of $\mathcal{A}$ over set $\mathcal{B}$ in the way described in lemma 1, matches with higher slackness are those that we can keep for a longer time while we move.

By reducing γ, we also reduce the number of matches that may be missed, although we also increase the computational cost of the algorithm. We will provide results to illustrate this tradeoff between efficiency and accuracy in section 6.

In this case testing every motion generated takes $O(n^{1.5} \log m)$ as we cannot take advantage of matching information for previously calculated cells. As we have $O(\frac{2\pi}{\gamma}^3 n^4m^4\lambda_q(nm))$ cells, then:

Lemma 4 *The computational cost of the approximate matching algorithm is* $O(\frac{2\pi}{\gamma}^3 n^5.5m^4\lambda_q(nm) \log n)$.

5 Lossless filtering preprocessing step

In most applications the cardinals of the two sets involved in matching problems are dissimilar and $|\mathcal{A}| << |\mathcal{B}|$. In general, algorithms are designed for sets of roughly equal cardinalities so they cannot take advantage of this situation if it occurs. We have adapted the Lossless Filtering algorithm presented in [5] to the three dimensional case. This algorithm discretizes the **NCPSM** problem by turning it into a series of smaller instances of itself and then solves them using the matching algorithm presented in Section 3. To achieve this discretization we use a conservative strategy that discards those subsets of $\mathcal{B}$ where no match may happen and keep a number of zones where this matches may occur. The discarding decisions throughout the first part of the process are made according to a series of geometric parameters that are invariant under rigid motion. These parameters help us to describe and compare the shapes of $\mathcal{A}$ and the different subsets of $\mathcal{B}$ that we explore. To navigate $\mathcal{B}$ and have easy access to its subsets, we use a *compressed octree* that is built using the points in set $\mathcal{B}$ as sites and completed with these geometric parameters. By doing this we achieve a reduction of the total computational time, corresponding to a pruning of the search space, as an effect of all the calculations we avoid by discarding parts of $\mathcal{B}$ cheaply and at an early stage.

The geometric parameters we use are: number of points, histogram of points'colors and maximum and minimum distance between points of every different color. Finally, as the candidate zones need not be located inside a certain node of the octree and may fall between various of its branches we need to perform diverse search function that take each possible case into account.

Finally, we state that this step does not increase the asymptotic cost of the algorithm, moreover, in section 6 we will show how it does produce a practical reduction in its running time.

Lemma 5 *The computational cost of the algorithm that combines the lossless filtering algorithm and the matching algorithm is* $O(\frac{2\pi}{\gamma}^3 n^5m^4\lambda_q(nm)) \log n$. *The bound is tight.*

6 Implementation and results

For the implementation we have used the C++ programming language under a Linux environment and the g++ compiler without compiler optimizations. All tests were run on a Pentium D machine with a 3 Ghz processor. Our implementation is still under development although we can already present some preliminary results.

For all the tests we begin with a specific realization of set $\mathcal{A}$ that has 10 points of three different colors. To build the various realizations of set $\mathcal{B}$ used, we applied a variable number of random rigid motions (noted Num_τ) to set $\mathcal{B}$ and a small perturbation to the point thus obtained. We completed set $\mathcal{B}$ by adding additional points in a variable proportion. We used a fixed value for parameter ϵ based on the *average inter point distance*. The values for parameter γ

are indicated in every case.

Effects of the lossless filtering algorithm. The performance of the algorithm depends on the effectiveness that the lossless Filtering step has on every data set, but at worst it meets the best (theoretical) running time up to date. In the best case, the initial problem is transformed into a series of subproblems of the same kind but with cardinality close to $n = |\mathcal{A}|$, producing a great saving of computational effort. In the following we try to quantify this saving in computational time. We compares the behavior of the Matching algorithm with and without the lossless filtering algorithm. The value used for gamma is 0.5.

The following table shows the cardinality of set $\mathcal{B}$ and the percentage of noise points (i.e. points that do not belong to any solution), the mean value of the cardinal of the set present in each candidate zone after the Lossless Filtering algorithm ($\overline{n'}$) the total time spent by the algorithm with Lossless Filtering preprocessing step (T1) and finally the total time spent by the algorithm without it (T2).

$\|\mathcal{B}\|$	% noise	$\overline{n'}$	T1(s)	T2(s)
120	66	13.25	8.9	903
250	80	24.4	113.3	8399.4
360	89	28.7	130.6	19290.4
400	87.5	20	106.6	38106
450	89	43.1	3000.9	54923.3
500	90	49.22	4955.6	73818.9

Concerning the mean value of n', we observe that the filtering algorithm manages to take away most of the complexity of the problem. Consequently, the main part of the computational effort can be attributed to the complexity inherent to the problem. Finally computational times increase with $|\mathcal{B}|$ although we believe that is kept to reasonable levels given the complexity of the calculations involved.

We must state that the sizes considered here are small given the huge computational costs of the algorithm without lossless filtering. The data presented shows that, even when the theoretical computational costs are still high due to the complexity inherent to the problem, using the lossless filtering results in significant computational saving.

Running time and accuracy study. In this test we seek to study the effect that parameter γ has on the number of solutions found and the time needed to find them. Amongst the values tested, we present two of the most extreme to illustrate the main tendencies that we observed. The table shows: The cardinal of set $\mathcal{B}$, de number of transformations of set $\mathcal{B}$ that it includes (Num_τ), the percentages of success in terms of the number of solutions found and the total times spent by the algorithm for different values of parameter γ.

$\|\mathcal{B}\|$	Num_τ	$\gamma = 2\pi$		$\gamma = 0.1$	
		succes%	Time(s)	*succes*%	Time(s)
150	5	80	33.9	100	85.8
270	9	100	39.5	100	793.7
360	12	91.6	401.4	100	1924.3
420	14	92.8	1123.9	92.8	2204.6
540	17	94.4	6020	100	11536.9
690	23	87	13690	95.6	28322.1

We observe that, as expected, bigger values of γ generally result in lower percentage of success. This happens because they may miss matchings with higher slackness. We also observe that smaller values of γ present higher running times of the algorithm. In this test we have increased the perturbation in every point in order to force the worst possible behavior of our algorithm by decreasing the slackness of the matchings to be found.

References

[1] C. Ambühl, S. Chakraborty, and B. Gartner. Computing largest common point sets under approximate congruence. *In Proc. 8th ESA, LNCS* 1879, pages 52-63, 2000.

[2] H. Alt, K. Mehlhorn, H. Wagener and E. Welzl. Congruence, similarity and symmetries of geometric objects. *Discrete & Computational Geometry*, 3:237–256, 1988.

[3] B. Chazelle, H. Edelsbrunner, L.J. Guibas and M. Sharir, "A singly-exponential stratification scheme for real semi-algebraic varieties and its applications", *Proc. 16th Internat. Colloq. Automata Lang. Program., LNCS 372, Springer-Verlag*, pp 179-192, 1989.

[4] V. Choi, N. Goyal. A Combinatorial Shape Matching Algorithm for Rigid Protein Docking. *CPM 2004, LNCS 3109*, pp. 285-296, 2004.

[5] Y. Diez, J.A. Sellarès. Efficient Colored Point Set Matching Under Noise *ICCSA 2007, LNCS 4705*, pp. 26-40, Springer-Verlag, 2007.

[6] A. Efrat, A. Itai, M. J. Katz, Geometry helps in Bottleneck Matching and related problems. *A*lgorithmica 31: 1-28, 2001.

[7] M. Sharir, "Recent Developments in Theory of Arrangements of Surfaces", *L*NCS, Vol 1738, pp 1-21, Springer-Verlag, 2000.

[8] M. Sharir and P.K. Agarwal, Davenport-Schinzel sequences and their geometric aplications, *C*ambridge University Press, 1995.

Pareto Envelopes in Simple Polygons*

Victor Chepoi† Karim Nouioua† Edouard Thiel† Yann Vaxès†

Abstract

For a set T of n points in a metric space (X, d), a point $y \in X$ is *dominated* by a point $x \in X$ if $d(x,t) \leq d(y,t)$ for all $t \in T$ and there exists $t' \in T$ such that $d(x,t') < d(y,t')$. The set of non-dominated points of X is called the *Pareto envelope* of T. H. Kuhn (1973) established that in Euclidean spaces, the Pareto envelopes and the convex hulls coincide. Chalmet et al. (1981) characterized the Pareto envelopes in the rectilinear plane $(\mathbb{R}^2, d_1)$ and constructed them in $O(n \log n)$ time. In this note, we investigate the Pareto envelopes of point-sets in simple polygons P endowed with geodesic d_2- or d_1-metrics (i.e., Euclidean and Manhattan metrics). We show that Kuhn's characterization extends to Pareto envelopes in simple polygons with d_2-metric, while that of Chalmet et al. extends to simple rectilinear polygons with d_1-metric. These characterizations provide efficient algorithms for construction of these Pareto envelopes.

1 Introduction

Convex hulls, in particular convex hulls in 2- and 3-dimensional spaces, are used in various applications and represent a basic object of investigations in computational geometry. They host such remarkable points as center, barycenter, and median as well as the optimal solutions of some NP-hard problems like the Steiner tree, the p-median, and the p-center problems. H. Kuhn [13] noticed that $\mathrm{conv}(T)$ can be described in truly distance terms: a point $p \in \mathbb{R}^m$ belongs to $\mathrm{conv}(T)$ if and only if the vector of Euclidean distances of p to the points of T is not dominated by the distance vector of any other point of $\mathbb{R}^m$. Inspired by this characterization of $\mathrm{conv}(T)$, one can define analogous geometric objects by replacing the Euclidean distance d_2 by any other distance d on $\mathbb{R}^m$, or by replacing $\mathbb{R}^m$ by a polygonal or a polyhedral domain endowed with an intrinsic distance. This leads to the following general concept of Pareto envelope. Given a set T of n points in a metric space (X, d), a point $y \in X$ is *dominated* by a point $x \in X$ if $d(x,t) \leq d(y,t)$ for all $t \in T$ and there exists $t' \in T$ such that $d(x,t') < d(y,t')$. The set of non-dominated points of X is called the *Pareto envelope* of T and is denoted by $\mathcal{P}_d(T)$.

Pareto envelopes have been investigated in several papers under the name of "sets of efficient points". Thisse, Ward, and Wendell [17] proved that $\mathcal{P}_{d_2}(T) = \mathrm{conv}(T)$ holds for all distances induced by round norms. The investigation of Pareto envelopes for particular polyhedral norms has been initiated by Wendell, Hurter, Lowe [21] and continued by Chalmet, Francis, Kolen [2] and Durier, Michelot [6, 7]. The main result of [2] is the following nice characterization of Pareto envelopes in the Manhattan plane:

$$\mathcal{P}_{d_1}(T) = \cap_{i=1}^{n}(\cup_{j=1}^{n} I_{d_1}(t_i, t_j)), \qquad (1)$$

where $I_{d_1}(t_i, t_j)$ is the smallest axis-parallel rectangle with diagonal $[t_i, t_j]$. This result was used in [2] to establish the correctness of an optimal $O(n \log n)$ sweeping-line algorithm for constructing $\mathcal{P}_{d_1}(T)$ in $\mathbb{R}^2$. Consequently, Pelegrin and Fernandez [14] described an algorithm for constructing Pareto envelopes in the plane endowed with a polygonal norm. Recently, Chepoi and Nouioua [5] characterized $\mathcal{P}_{d_1}(T)$ in $(\mathbb{R}^3, d_1)$ and showed that the characterization of Chalmet et al. [2] holds for $\mathcal{P}_{d_\infty}(T)$ in $(\mathbb{R}^m, d_\infty)$. They also presented efficient algorithms for constructing $\mathcal{P}_{d_1}(T)$ and $\mathcal{P}_{d_\infty}(T)$ in $\mathbb{R}^3$. We refer to [5] for other references on Pareto envelopes in normed spaces and their applications.

In this note, we characterize and efficiently construct the Pareto envelopes of sets in simple polygons endowed with the geodesic d_2 and d_1-distances. Distance problems for simple polygons constitute a classical subject in computational geometry; [9, 11, 15, 16, 18] is a small sample of papers devoted to this subject. We show that, like in Euclidean spaces, Pareto envelopes of finite sets in simple polygons with d_2-distance coincide with their geodesic convex hulls and therefore can be constructed using an algorithm of Toussaint [18]. On the other hand, we show that Pareto envelopes in simple rectilinear polygons can be characterized using equality (1). This characterization is used to design an efficient algorithm for constructing these envelopes. Due to space constraints, the proofs of several results in last section are postponed to the full version.

We conclude this section with some definitions. Let (X, d) be a metric space. The *interval* $I(x, y)$ between two points $x, y \in X$ consists of all points *between* x

*This research was partly supported by the ANR grant BLAN06-1-138894 (projet OPTICOMB).

†LIF, Faculté des Sciences de Luminy, Université de la Méditerranée, F-13288 Marseille Cedex 9, France, {chepoi,nouioua,thiel,vaxes}@lif.univ-mrs.fr

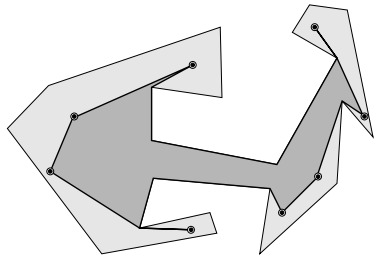


Figure 1: Example of $\mathcal{P}_{d_2}(T)$

and y: $I(x,y) := \{u \in X : d(x,u)+d(u,y) = d(x,y)\}$. A set M of X is *convex* if $I(x,y) \subseteq M$ for all $x, y \in M$. The *convex hull* $\mathrm{conv}(S)$ of a set $S \subset X$ is the smallest convex set containing S.

2 Simple polygons

In this section, P is a simple polygon with m sides endowed with the geodesic d_2-metric. For two points $x, y \in P$, $\gamma(x,y)$ is the unique geodesic path inside P between x and y, and $d_2(x,y)$ is the length of this path. For a set of n points $T \subset P$, we denote by $\mathrm{conv}(T)$ and $\mathcal{P}_{d_2}(T)$ the *geodesic convex hull* and the Pareto envelope of T. Since two points of a simple polygon P are connected by a unique geodesic, (P, d_2) is a metric space of global non-positive curvature, i.e. a *CAT(0)-space* [1]. CAT(0) spaces are characterized in several ways (in particular, by uniqueness of geodesic paths, convexity of the distance function, etc.) and have many important properties, placing them in the center of modern geometry; for results and definitions the reader can consult the book [1]. Below we will show that $\mathcal{P}_d(T) \subseteq \mathrm{conv}(T)$ holds for any finite subset of a CAT(0)-space (X, d) and we conjecture that in fact $\mathcal{P}_d(T) = \mathrm{conv}(T)$ holds.

2.1 $\mathcal{P}_{d_2}(T) = \mathrm{conv}(T)$

We aim to establish the following result:

Proposition 1 *$\mathcal{P}_{d_2}(T) = \mathrm{conv}(T)$. Consequently, $\mathcal{P}_{d_2}(T)$ can be constructed in $O(m + n \log m)$-time.*

The inclusion $\mathcal{P}_{d_2}(T) \subseteq \mathrm{conv}(T)$ follows from the following more general result:

Lemma 1 *$\mathcal{P}_d(T) \subseteq \mathrm{conv}(T)$ for any finite set of a CAT(0) metric space (X, d).*

Proof. Let $x \notin \mathrm{conv}(T)$. By Proposition 2.4(1) of [1] there exists a unique point $\pi(x)$ (the metric projection of x) such that $d(x, \pi(x)) = \inf_{y \in \mathrm{conv}(T)} d(x,y)$. As in the case of Euclidean spaces, $\pi(x)$ can be viewed as the orthogonal projection of x on $\mathrm{conv}(T)$, because by Proposition 2.4(3) the Alexandrov angle α at $\pi(x)$ between the geodesics $\gamma(x, \pi(x))$ and $\gamma(y, \pi(x))$ is at least $\pi/2$ for any point $y \in \mathrm{conv}(T), y \neq \pi(x)$. By law of cosines which holds in CAT(0) spaces (page 163 of [1]), if $a = d(x, \pi(x)), b = d(y, \pi(x))$, and $c = d(x,y)$, then $c^2 \geq a^2 + b^2 - 2ab\cos\alpha \geq a^2 + b^2 > b^2$ for any $y \in \mathrm{conv}(T), y \neq \pi(x)$. Hence $d(x,y) > d(\pi(x), y)$, i.e., x is dominated by $\pi(x)$. Since x is an arbitrary point outside $\mathrm{conv}(T)$, this implies that $\mathcal{P}_d(T) \subseteq \mathrm{conv}(T)$. □

Now we show the converse inclusion $\mathrm{conv}(T) \subseteq \mathcal{P}_{d_2}(T)$. Pick $q \in \mathrm{conv}(T)$. If q belongs to the boundary of $\mathrm{conv}(T)$, then q belongs to the geodesic path $\gamma(t, t')$ between two vertices t, t' of $\mathrm{conv}(T)$. Since $t, t' \in T$, if q is dominated by some point p, then $d_2(p,t) \leq d_2(q,t)$ and $d_2(p,t') \leq d(q,t')$. Since $q \in \gamma(t,t')$, this is possible only if these inequalities hold as equalities, thus $p \in \gamma(t,t')$, yielding $p = q$. Thus $q \in \mathcal{P}_{d_2}(T)$ in this case. Now, suppose that q belongs to the interior of the simple polygon $\mathrm{conv}(T)$. Suppose by way of contradiction that q is dominated by some point $p' \in P$. By Lemma 1 of [15] the distance function d_2 on P is convex. This means that for any point $t \in T$, as p varies along the geodesic $\gamma(p', q)$, $d_2(t,p)$ is a convex function of p. Since q belongs to the interior of $\mathrm{conv}(T)$, one can select a point $p \in \gamma(p', q) \cap \mathrm{conv}(T)$ which still dominates q and is visible from q (i.e., $[p,q] \subseteq P$). Denote by q' the first intersection of the boundary of $\mathrm{conv}(T)$ with the ray with origin p which passes via the point q. By the definition of q', we infer that $q \in [p, q'] = \gamma(p, q')$. Pick any point $t \in T$. By second part of Lemma 1 of [15], $d_2(t,q) < \max\{d_2(t,q'), d_2(t,p)\}$. Since $d_2(t,p) \leq d_2(t,q)$ by the choice of p, we obtain that $d_2(t,q) < d_2(t,q')$. Since this inequality holds for all points of T, q and p both dominate the boundary point q', a contradiction with $q' \in \mathcal{P}_{d_2}(T)$.

G. Toussaint [18] presented an $O(m+n\log m)$-time algorithm for constructing the geodesic convex hull of an n-point set T of a simple polygon P with m sides. Together with Proposition 1 this shows that $\mathcal{P}_{d_2}(T)$ can be constructed within the same time bounds.

3 Simple rectilinear polygons

In this section, P is a simple rectilinear polygon (i.e., a simple polygon having all edges axis–parallel) with m edges endowed with the geodesic d_1-metric.. A *rectilinear path* is a polygonal chain consisting of axis–parallel segments lying inside P. The length of a rectilinear path in the d_1-metric equals the sum of the lengths of its constituent segments. For two points $x, y \in P$, the *geodesic d_1-distance* $d_1(x,y)$ is the length of the minimum length rectilinear path (i.e., rectilinear geodesic) connecting x and y. An axis–parallel segment c is a *cut segment* of P if it connects two edges of P and lies entirely in P. One basic property of resulting metric space (P, d_1) is that its axis-parallel cuts and the two subpolygons defined by such cuts are convex and gated [4]. A subset M of a metric space (X, d) is called *gated* [19] provided every point $v \in X$ admits a *gate* in M, i.e. a point $g(v) \in M$ such that

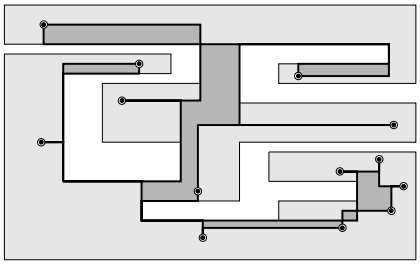


Figure 2: Example of $\mathcal{P}_{d_1}(T)$

$g(v) \in I(v,u)$ for all $u \in M$.

3.1 Characterization

We extend the characterization of [2] to Pareto envelopes $\mathcal{P}_{d_1}(T)$ in rectilinear polygons:

Proposition 2 $\mathcal{P}_{d_1}(T) = \cap_{i=1}^{n}(\cup_{j=1}^{n} I(t_i,t_j))$.

Proof. One direction of the proof is obvious: if p is not Pareto, then $p \notin \cup_{j=1}^{n} I(t_i,t_j)$. To prove the converse, let $p \in \mathcal{P}_{d_1}(T)$ but $p \notin \cup_{j=1}^{n} I(t_i,t_j)$ for some t_i. Let $c_v = [q',q'']$ and $c_h = [p',p'']$ be the maximal vertical and horizontal cuts which pass through the point p. Denote by P_1, P_2, P_3 and P_4 the subpolygons of P defined by these cuts. Let $P_1 \cap P_3 = P_2 \cap P_4 = \{p\}$ and $t_i \in P_1$. Obviously, $P_1, \ldots, P_4$ are gated. Note that p is the gate in P_1 of any point of P_3. As $p \notin \cup_{j=1}^{n} I(t_i,t_j)$, we conclude that $P_3 \cap T = \emptyset$. Set $P_{j,k} := P_j \cup P_k$, where $j,k \in \{1,2,3,4\}$ and $j \neq k$. Note that the four subpolygons $P_{j,j+1(\mathrm{mod}\, j)}$ are gated sets of P as intersection of gated sets. We distinguish two cases: (i) p is the gate of t_i in one of the cuts c_h or c_v, say the first, and (ii) the gates q and z of t_i in c_v and c_h are different from p.

First, consider the case (i). Since p is the gate of t_i in c_h, obviously it is also the gate of t_i in $P_{3,4}$. From the choice of p and t_i we conclude that $P_{3,4} \cap T = \emptyset$. Let $g_1, \ldots, g_n$ be the gates of $t_1, \ldots, t_n$ of T in c_v. First, suppose that these gates are all different from p. Then all $g_1, \ldots, g_n$ belong to the segment $[q',p] \subset c_v$ which separates P_1 and P_2. Let g_k be the closest to p such gate. Then $g_k \in I(p,g_j)$ and, since $g_j \in I(p,t_j)$, we infer that $g_k \in \cap_{j=1}^{n} I(p,t_j)$, thus g_k dominates p, contradiction that p is Pareto. Now assume that p is the gate of some point $t_j \neq t_i$ in c_v. If $t_j \in P_2$, then p is the gate of t_j in $P_{1,4}$, contrary to $p \notin I(t_i,t_j)$. Thus $t_j \in P_1$. Let u and w be the gates of t_i and t_j in c_v and c_h. Pick some rectilinear geodesics $\gamma(t_i,u), \gamma(t_j,w)$, and $\gamma(t_i,t_j)$ between the pairs t_i,u; t_j,w, and t_i,t_j, respectively. Since $p \notin I(t_i,t_j)$, $\gamma(t_i,t_j)$ cannot share common points with both segments $[u,p]$ and $[p,w]$. Let $\gamma(t_i,t_j) \cap [u,p] = \emptyset$. Let u' be a closest to u point of $\gamma(t_i,u) \cap \gamma(t_i,t_j)$. Necessarily $u' \neq u$. Let w' be a closest to w point of $\gamma(t_j,w) \cap \gamma(t_i,t_j)$. Since P is a simple polygon, the region of the plane bounded by $[u,p], [p,w]$, the part of $\gamma(t_i,u)$ between u,u', the part of $\gamma(t_i,t_j)$ between u',w', and the part of $\gamma(t_j,w)$ between w',w, is contained in P. Let $[u'',u]$ be the last link in the subpath of $\gamma(t_i,u)$ between u' and u. Then for some $\delta > 0$, the segment $[v',v'']$ belongs to P, where $v' \in [u'',u], v'' \in [p,w]$ and $d(u,v') = d(p,v'') = \delta$. This contradicts that p is the gate of t_i in c_h.

Now, consider case (ii). Let u be the furthest from t_i point of $I(t_i,q) \cap I(t_i,z)$. Pick rectilinear geodesics $\gamma(u,q)$ and $\gamma(u,z)$ between u,q and u,z. Let $[q',q]$ and $[z',z]$ be the last links of these paths. Let q'' be the point of c_h with the same x-coordinate as q'. Let z'' be the point of c_v with the same y-coordinate as z'. Since P is a simple polygon, the region between $[q,p], [z,p]$ and $\gamma(u,q), \gamma(u,z)$ belongs to P. Moreover, since $q,z \in I(p,t_i)$ and $I(p,t_i)$ is convex, this region necessarily belongs to $I(p,t_i)$. In particular, both rectangles $R' = [q',q,p,q'']$ and $R'' = [z',z,p,z'']$ belong to $I(p,t_i)$. As we already stated, all points of T are outside P_3. Let g_v be the closest to p gate in c_v of a point of $T \cap P_2$, while g_h be the closest to p gate in c_h of a point of $T \cap P_4$. Since $p \notin \cup_{j=1}^{n} I(t_i,t_j)$, we conclude that g_v and g_h are different from p. Let $0 < \delta < \min\{d(p,g_v), d(p,g_h), d(z',z), d(q',q)\}$. Consider a point $p' \in R' \cap R''$ whose coordinates differ by δ from those of p. Since $p' \in I(p,t_i)$, we obtain that $d(p',t_i) = d(p,t_i) - 2\delta$. For any other t_j we have $d(p',t_j) \leq d(p,t_j)$. This contradicts that p is Pareto. □

A subset S of P is *ortho-convex* if the intersection of S with any axis–parallel cut of P is connected.

Lemma 2 *$\mathcal{P}_{d_1}(T)$ is a closed ortho-convex set of P.*

3.2 The algorithm

Now, we describe the algorithm for constructing the Pareto envelope $\mathcal{P}_{d_1}(T)$ for a set T of n points in a simple rectilinear polygon P with m vertices. In the sequel, we will refer to points of T as *terminals*. The algorithm uses Chazelle's algorithm for computing all vertex-edge visible pairs of a simple polygon [3] and the optimal point-location methods [8, 12]. Using Chazelle's algorithm, we derive a decomposition of the polygon P into rectangles, employing only horizontal cuts which pass through the vertices of P. Using the optimal point-location methods [8, 12] we compute in $O(n \log m)$ total time which rectangles of the decomposition contain the terminals (notice that the induced subdivision is monotone, hence the point-location structure can be built in linear time). At the next step, we sort by y all terminals from each rectangle. With these sorted lists, we refine the initial subdivision by dividing each rectangle containing terminals with the horizontal cuts passing via terminals. The dual graph of this decomposition $\mathcal{D}$ is a tree $\mathcal{T}$:

the nodes of a tree are the rectangles of $\mathcal{D}$, and two nodes in $\mathcal{T}$ are adjacent iff the corresponding rectangles are bounded by the common cut. We suppose that $\mathcal{T}$ is rooted at some rectangle. Any cut c of our subdivision divides the polygon P into two subpolygons P'_c and P''_c which correspond to two subtrees $\mathcal{T}'_c$ and $\mathcal{T}''_c$ of $\mathcal{T}$. It can be easily shown that if $P''_c \cap T = \emptyset$ (in this case we say that P''_c is T-empty), then $\mathcal{P}_{d_1}(T)$ is contained in $P'_c \cup c$ (any point of P''_c is dominated by its gate in c). By proceeding the tree $\mathcal{T}$, in linear time we can remove all T-empty subpolygons and their corresponding subtrees. We will denote the resulting polygon, subdivision, and tree also by $P, \mathcal{D}$, and $\mathcal{T}$. The resulting decomposition $\mathcal{D}$ and its tree $\mathcal{T}$ can be constructed in time $O(m + n(\log n + \log m))$. If all terminals are vertices of P, then we avoid the application of point-location methods and ranking of terminals, requiring only $O(n + m)$ time.

Given a non-root rectangle R, we denote by e'_R and e''_R the horizontal sides of R, so that e'_R separates R from to the root of $\mathcal{T}$. The set of gates of all terminals in R can be partitioned into the subset G'_R of gates located on e'_R and the subset G''_R of gates located on e''_R. Let $g'_l(R), g'_r(R)$ be the leftmost and the rightmost points from G'_R and let $g''_l(R)$, $g''_r(R)$ be the leftmost and the rightmost points from G''_R. In the full version, we show how to compute the four extremal gates $g'_l(R)$, $g'_r(R)$ $g''_l(R)$ and $g''_r(R)$ for all rectangles $R \in \mathcal{D}$ in total linear time by using an upward and a downward traversal of $\mathcal{T}$.

For each rectangle $R \in \mathcal{D}$, given the quadruplet of gates $Q_R = \{g'_l(R), g'_r(R), g''_l(R), g''_r(R)\}$, at the next step we compute the Pareto envelope $\mathcal{P}_{d_1}(Q_R)$ of Q_R. It consists of a *box* B_R having its horizontal sides on the sides of R and two horizontal segments which are incident either to two points of the quadruplet lying on the same horizontal side of R or to two opposite points lying on different horizontal sides of R (one or both these segments can be degenerated). In general, these segments do not necessarily belong to the final Pareto envelope $\mathcal{P}_{d_1}(T)$. On the other hand, as we will show below, B_R minus its horizontal sides is exactly the set $\mathcal{P}_{d_1}(T) \cap R^0$, where $R^0 := R \setminus (e'_R \cup e''_R)$ (clearly, the horizontal sides of B_R belong to $\mathcal{P}_{d_1}(T)$ as well because $\mathcal{P}_{d_1}(T)$ is closed). Now, if we consider any horizontal cut c, then we show that $\mathcal{P}_{d_1}(T) \cap c$ is the smallest segment $s_c \subseteq c$ spanned by the terminals and/or the horizontal sides of all boxes B_R located on c. Clearly, having at hand the four gates of each rectangle, the sets B_R and s_c can be determined in $O(n + m)$ time. To conclude, it remains to prove the correctness of two last steps of the algorithm. This follows from the following two lemmata whose proof is given in the full version.

Lemma 3 $\mathcal{P}_{d_1}(T) \cap R^0 = B_R \cap R^0$.

Lemma 4 $\mathcal{P}_{d_1}(T) \cap c = s_c$.

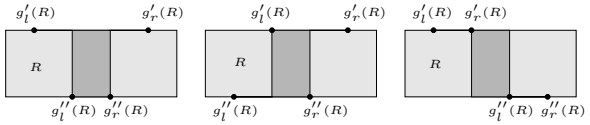


Figure 3: $\mathcal{P}_{d_1}(Q_R)$

Summarizing the results of this section, we obtain our main result:

Theorem 5 *The Pareto envelope of n terminals located in a simple rectilinear polygon P with m edges can be constructed in time $O(n + m(\log n + \log m))$ ($O(n + m)$ if all terminals are vertices of P).*

References

[1] M. Bridson, A. Haefliger, *Metric Spaces of Non-Positive Curvature,* Springer, Berlin, 1999.

[2] G. Chalmet, L. Francis, and A. Kolen, Finding efficient solutions for rectilinear distance location problems efficiently, *Europ. J. Oper. Res.* **6** (1981), 117–124.

[3] B. Chazelle, Triangulating a simple polygon in linear time, *Discrete Comput. Geom.*, **6**(1991), 485–524.

[4] V. Chepoi and F.F. Dragan, Computing a median point of a simple rectilinear polygon, *Inform. Process. Lett.* **49**(1994), 281–285.

[5] V. Chepoi and K. Nouioua, Pareto envelopes in $\mathbb{R}^3$ under l_1 and l_∞-distance functions, *Symposium on Computational Geometry* 2007, 284–293.

[6] R. Durier, On Pareto optima, the Fermat-Werber problem, and polyhedral gauges. *Math. Progr.* **47** (1990), 65–79.

[7] R. Durier and C. Michelot, Sets of efficient points in normed space, *J. Math. Analys. Appl.* **117** (1986), 506–528.

[8] H. Edelsbrunner, L.J. Guibas, and J. Stolfi, Optimal point location in a monotone subdivision, *SIAM J. Comput.*, **15**(1985), 317–340.

[9] L. J. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan, Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons, *Algorithmica* **2** (1987), 209–233.

[10] P. Hansen, T. Perreur, and J.F. Thisse, Location theory, dominance and convexity : some further results. *Oper. Res.* **28** (1980), 1241–1250.

[11] J. Hershberger and S. Suri, Matrix searching with the shortest-path metric, *SIAM J. Comput.* 26(6), 1612-1634 (1997).

[12] D.G. Kirkpatrick, Optimal search in planar subdivisions, *SIAM J. Comput.*, **12**(1983), 28–35.

[13] H. W. Kuhn, A note on Fermat's problem, *Math. Prog.* **4** (1973), pp. 98–107.

[14] B. Pelegrin and F.R. Fernandez, Determination of efficient points in multiple-objective location problems, *Nav. Res. Logistics* **35** (1988), 697–705.

[15] R. Pollack, M. Sharir, and G. Rote, Computing the geodesic center of a simple polygon, *Discr. Comput. Geom.* **4** (1989), 611–626.

[16] S. Suri, Computing geodesic furthest neighbors in simple polygons, *J. Comput. Syst. Sci.* **39** (1989) 220–235.

[17] J. F. Thisse, J. E. Ward, and R. E. Wendell, Some properties of location problems with block and round norms. *Oper. Res.* **32** (1984), 1309–1327.

[18] G. T. Toussaint, Computing geodesic properties inside a simple polygon, *Revue D'Intelligence Artificielle,* **3** (1989), 9–42.

[19] M. van de Vel, *Theory of Convex Structures*, Elsevier, Amsterdam, 1993.

[20] R. E. Wendell and A.P. Hurter, Location theory, dominance, and convexity. *Oper. Res.* **21** (1973), 314–320.

[21] R. E. Wendell, A.P. Hurter, and T.J. Lowe, Efficient points in location theory. *AIEE Trans.* **9** (1973), 238–246.

Shortest Inspection-Path Queries in Simple Polygons

Christian Knauer* Günter Rote* Lena Schlipf*

Abstract

We want to preprocess a simple n-vertex polygon P to quickly determine the *shortest* path p from a fixed source point $s \in P$ to view a set $Q \subseteq P$ of query points (i.e., such that each point $q \in Q$ is visible from some point on the path p). We call such queries *shortest inspection-path queries*. For $|Q| \leqslant 2$ we describe data structures that answer such queries in logarithmic time. The structures have linear (for $|Q| = 1$) respectively quadratic (for $|Q| = 2$) size and preprocessing time.

1 Introduction

Many variations of the problem of computing shortest paths in *simple* polygons have been studied in the past, c.f. [1]. One instance of the problem is to find the shortest path from a given fixed source point s in a simple polygon P with n vertices to view a set $Q \subseteq P$ of query points. Our goal is to preprocess the input (P, s) to answer queries of this type: Given a set $Q \subseteq P$ of query points, find the shortest distance one needs to travel in P from s to see all points in Q. For $|Q| = 1$ the query can be answered in $O(n)$ time without preprocessing [6], and in $O(\log n)$ time with $O(n^2)$ preprocessing time and space [7]. We improve and simplify the latter result and describe a solution with linear preprocessing time and space that achieves $O(\log n)$ query time. For $|Q| = 2$ we describe a solution with quadratic preprocessing time and space that also achieves logarithmic query time. The results are summarized in the following

Theorem 1 *Let P be a simple polygon with n vertices and let $s \in P$ be a fixed point. We can preprocess (P, s)*

- *in $O(n)$ time into a data structure of $O(n)$ size that can answer shortest inspection-path queries for $Q \subseteq P$ with $|Q| = 1$ in $O(\log n)$ time, and*
- *in $O(n^2)$ time into a data structure of $O(n^2)$ size that can answer shortest inspection-path queries for $Q \subseteq P$ with $|Q| = 2$ in $O(\log n)$ time.*

*Institut für Informatik, Freie Universität Berlin, Takustraße 9, D–14195 Berlin, Germany. E-mail: {knauer,rote,schlipf}@inf.fu-berlin.de

Preliminaries. The visibility polygon of a point $q \in P$ will be denoted by $V(q)$, the shortest path in P between two points $x, y \in P$ by $p(x, y)$ and the shortest path tree from s in P by T_s. For a shortest path $p = p(x, y)$ we denote by $\hat{x}$ (resp. $\hat{y}$) the *first vertex of P on p after x* (resp. the *last vertex of P on p before y*). If we remove $V(q)$ from P, the polygon splits into disconnected regions that we call *invisible regions*. Each such region has exactly one edge in common with $V(q)$. By $P_s(q)$ we denote the region which contains s. The common edge $w(q)$ between $V(q)$ and $P_s(q)$ will be called *the window of q*. Let a and b be the endpoints of $w(q)$, and r be the last common vertex between the two paths $p(s, a)$ and $p(s, b)$ (i.e., the lowest common ancestor $LCA_{T_s}(a, b)$ of a and b in T_s), cf. Fig. 1. The paths $p(r, a)$ and $p(r, b)$ together with the segment $w(q)$ form the *funnel of q* which will be denoted by $F(q)$, cf. Fig. 2; the vertex r is called the *root* of the funnel, the segment $w(q)$ is called the *base* of the funnel. Note that the paths $p(r, a)$ and $p(r, b)$ are outward convex. In Section 3 we require a generalization of the notion of a funnel which was introduced in [3]: The *hourglass* between two line segments $l_1, l_2 \subseteq P$ is the boundary of the union of all shortest paths $p(x, y)$ for $x \in l_1$, $y \in l_2$; it will be denoted by $H(l_1, l_2)$. For two paths p, q we denote the concatenation of p and q by $p + q$.

In Section 2 we first give a structural characterization of the solution that forms the basis of our approach and then prove the case $|Q| = 1$ of Theorem 1. We consider the case $|Q| = 2$ in Section 3 and provide some conclusions in Section 4. Due to space constraints we omit most of the technical details from this abstract; they can be found in [8, 10].

2 The data structure for one query point

If $Q = \{q\}$ we have to find a point $c \in P$ visible from the query point $q \in P$ that has the shortest distance from s. If q is invisible from s, then s lies in an invisible region (if q is visible from s, then clearly $c = s$). In this case it is easy to see that the point c lies on the window $w(q)$, in particular c is the point on $w(q)$ that has the shortest distance to s, cf. Fig. 1. A simple characterization of c was given in [7]: Let $a = v_0, v_1, \ldots, r = v_m, \ldots, v_k, v_{k+1} = b$ denote the vertices of the funnel from a to b. $F(q)$ can be decomposed into triangles by extending the edges of $F(q)$ until they intersect $w(q)$. Let x_i denote the intersec-

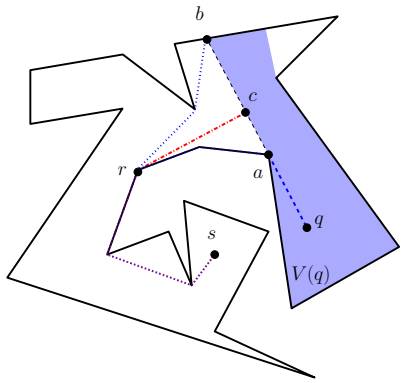


Figure 1: The window $w(q) = ab$ separates $P_s(q)$ and $V(q)$. The point with shortest distance to s that is visible from q is c. The drawing shows $p(s, a)$, $p(s, b)$ and $p(s, c)$.

tion point of the extension of the edge v_iv_{i+1} with $w(q)$ (hence, $x_0 = a$ and $x_k = b$). The shortest path from s to points on the segment x_ix_{i+1} passes through v_i as the last vertex of P. Denote the angles between the extension edges and the window by $\theta_0, \theta_1, \ldots, \theta_k$, i.e., $\theta_i = \angle bx_iv_i$ for $0 \leq i < k$ and $\theta_k = \pi - \angle abv_k$. The outward convexity of the paths $p(r, a), p(r, b)$ im-

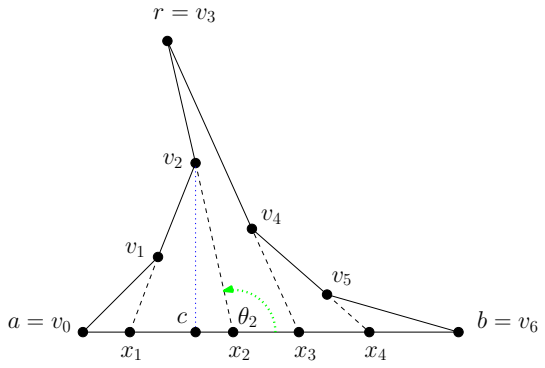


Figure 2: The funnel $F(q)$ over the window $w(q) = ab$. The optimal point c is the foot of the perpendicular from v_2 to $w(q)$.

plies that the sequence $\theta_0, \theta_1, \ldots, \theta_k$ is increasing. The optimal contact point c can now easily be characterized in terms of the sequence $\theta_0, \theta_1, \ldots, \theta_k$, e.g., in the case where $\theta_i < \pi/2$ and $\theta_i + 1 > \pi/2$ for some $0 \leq i \leq k$, c is the foot of the perpendicular from v_{i+1} to $w(q)$. We can therefore search for c by looking at the angles θ_i: If $\theta_i > \pi/2$ then c lies left of x_i, whereas if $\theta_i < \pi/2$ then c lies right of x_i.

To answer a query q we will proceed in two steps: First we compute the window $w(q)$ of q along with the funnel root r. Then we compute the optimal point c on $w(q)$.

After the preprocessing phase, the first step and the second step can be done in $O(\log n)$ time.

2.1 Preprocessing phase

1. Store the vertices of P in an array A, sorted in clockwise order along the boundary of P.
2. Compute a data structure $\mathcal{D}_1$ that supports $O(\log n)$ time shortest path queries in P between any pair of points $u, t \in P$ [2].
3. Compute a data structure $\mathcal{D}_2$ that supports $O(\log n)$ time ray-shooting queries to P [5].
4. Compute the shortest path tree T_s [3] and preprocess it to support $O(1)$ time LCA-queries [4].

The total preprocessing time and space is $O(n)$.

2.2 Query phase

In the query phase we will check at first if q is visible from s (in this case $c = s$). This can be done in $O(\log n)$ time by shooting a ray from q in the direction of s and testing if the boundary of P is hit before s. In the following we can assume that q is not visible from s.

Computing the funnel. To find $w(q) = ab$ and r in $O(\log n)$ time in the first step of the query phase we proceed as follows: Since the window separating s from $V(q)$ is specified by the last vertex of P on the shortest path from s to q (Fig. 1), we can find $a = \hat{q}$ in $O(\log n)$ time via $\mathcal{D}_1$. To find b in $O(\log n)$ time we shoot a ray from q in the direction of a. Next, we compute $v_k = \hat{b}$ in $O(\log n)$ time via $\mathcal{D}_1$, and finally, we get the funnel root $r = LCA_{T_s}(a, v_k)$ in $O(1)$ time.

Computing the optimal point on the window. To find the optimal point c on $w(q)$ in $O(\log n)$ time in the second step of the query phase we proceed as follows:

- First we check if $\theta_0 > \pi/2$ or $\theta_k < \pi/2$. In the first case $c = a$, in the second case $c = b$, and in either case we are finished.
- Next we look at the extensions of the edges emanating from the apex $r = v_m$ of the funnel. If $\theta_{m-1} = \pi/2$, or $\theta_m = \pi/2$, or $\theta_{m-1} \leq \pi/2 < \theta_m$, c is the foot of the perpendicular from v_m to $w(q)$ and we are finished.
- If $\theta_{m-1} > \pi/2$, then $\theta_i > \pi/2$ for $m \leq i \leq k$, since the angle sequence is increasing. In particular c is the foot of the perpendicular from some vertex v_i to $w(q)$, where v_i is on the left side $p(r, a)$ of the funnel $F(q)$, i.e., $1 \leq i < m$. To determine for which vertex v_i the perpendicular

to $w(q)$ has to be drawn, we would like to perform a binary search on the sequence $v_0, \ldots, v_k$. However this sequence is not directly accessible, so we use the array A instead, and perform a binary search on the interval $[r, a]$ in A (if $r = s$ and s is not a vertex from P, we take the next vertex $\hat{s}$ after s on $p(s, a)$ and search in the interval $[\hat{s}, a]$ instead). For a vertex u in this interval we compute $LCA_{T_s}(u, a)$, which is one of the vertices $v_0, \ldots, v_m$ on the left edge of the funnel, say v_i. By computing the angle θ_i we can decide if the binary search has to continue to the left or to the right of u. After $O(\log n)$ iterations the binary search is narrowed down to an interval between two successive vertices in A. This implies that the point v_i from which the perpendicular to c has to be drawn is also determined.

Note that for several successive vertices u_j in $[r, a]$ we can get the same vertex v_i as a result of computing $LCA_{T_s}(u_j, a)$. But the number of vertices in $[r, a]$ is $O(n)$ and so still after $O(\log n)$ iteration the binary search is narrowed down to an interval between two successive vertices in A.

- The case that $\theta_{m-1} < \pi/2$ is symmetric to the previous case.

In the end, we can compute the length of the shortest path in constant time from the information stored in T_s. The shortest path itself can be output in time linear in its length.

3 The data structure for two query points

There are several cases for the optimal path p if $Q = \{q_1, q_2\}$. We discuss here only the most interesting one: p first reaches $w_1 = w(q_1)$ to see q_1 where it is *reflected* and then proceeds to $w_2 = w(q_2)$ to see q_2 (or vice versa), c.f. Fig. 4 for an example (we will also restrict our attention here to the case where w_1 and w_2 do not cross). For a full discussion of all cases, like when w_1 and w_2 intersect, or when w_1 lies 'behind' w_2 (meaning that w_1 lies completely inside $P \setminus P_s(q_2)$), or the easy cases when q_1 or q_2 are already visible from s, we refer to [10].

The basic idea behind computing p is to (conceptually) reflect the polygon $P_1 = P_s(q_1)$ at the window w_1. The resulting polygon P_1' is then 'glued' to P_1 along w_1 (and the polygon $P \setminus P_1$ is discarded) to form a (possibly self-overlapping) polygon P_1^*. We then compute the shortest path p^* in P_1^* from s to see q_2', the reflection of q_2 at w_1.

To compute (the length of) p^* during a query, we (implicitly) determine the funnel F^* of q_2' in P_1^* and then compute the optimal point on the funnel base w_2' by binary search on its boundaries as in the previous section. We get F^* by combining the funnel $F_1 = F(q_1)$ and the hourglass $H_{12}' = H(w_1, w_2')$ using the technique of [2] (which is essentially a binary search). (The boundaries of) F_1 and H_{12}' (and thus of F^*) will be represented implicitly as paths in precomputed shortest-path trees.

Note that the reflection at w_1 is not performed explicitly, but 'on demand'.

3.1 Preprocessing phase

1. Compute A, $\mathcal{D}_1$, $\mathcal{D}_2$, and T_s as in Section 3.1.
2. For each vertex v of P compute the shortest path tree T_v and preprocess it to support $O(1)$ time LCA-queries.

The total preprocessing time and space is $O(n^2)$.

3.2 Query phase

As in Section 2.2 a query will be answered in two steps: First we compute the funnel F^* of q_2' in P_1^* (which represents the shortest paths in P from s to the points of w_2 which are reflected on w_1). Then we compute the optimal point c on w_2', the base of the funnel F^*.

Computing the funnel. As in Section 2.2 we first use $\mathcal{D}_1$ and T_s to compute in $O(\log n)$ time the windows $w_1 = a_1 b_1$ and $w_2 = a_2 b_2$, and the root r of the funnel F_1. The hourglass H_{12} (which is the 'unreflected' version of H_{12}') consists of the segments w_1 and w_2 together with the two paths $p(a_1, a_2)$ and $p(b_1, b_2)$ (which are mirrored at the line through w_1).

We then compute F^* by combining the funnel F_1 and the hourglass H_{12}' using the technique of [2]. To this end we need to construct the four common tangents $t_1, \ldots, t_4$ that touch one side from F_1 or H_{12}'. Since we cannot afford to reflect P_1 explicitly at w_1 during a query we compute F^* as the combination of F_1 with H_{12} instead. To this end the 'tangents' $t_1, \ldots, t_4$ have to be 'folded' at w_1, cf. Fig 3. Each tangent can be found in $O(\log n)$ time by performing a binary search on the vertices of $p(r, a_1)$, $p(r, b_1)$, $p(a_1, a_2)$ and $p(b_1, b_2)$ [9]. These vertices are not directly accessible but given only implicitly via the trees $T_{\hat{r}}$, $T_{\hat{a}_1}$, and $T_{\hat{b}_1}$ (we can compute $\hat{r}$, $\hat{a}_1$ and $\hat{b}_1$ using $\mathcal{D}_1$ in $O(\log n)$ time). Therefore we have to perform the binary search on the array A instead (as in the previous section).

In the example depicted in Fig. 4 the sides of F^* are $p(r, a_2)' = p(r, m_1) + m_1 m_2^* + p(m_2, a_2)$ and $p(r, b_2)' = p(r, m_3) + m_3 m_4^* + p(m_4, b_2)$ where m_1 is the last point on $p(r, a_1)$ which belongs also to a side of F^* and m_2 is the first point on $p(a_1, a_2)$ which belongs also to a side of F^*. The same holds for m_3 and m_4. The points m_1, m_2, m_3 and m_4 are the points on which F_1 and H_{12} are connected, so they can be obtained in the same way as in [2].

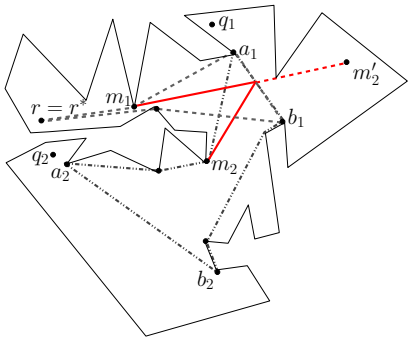


Figure 3: To obtain m_2' we reflect m_2 on w_1. We construct the line l_1 between m_2' and m_1 and reflect l_1 on w_1. Let $m_1m_2^*$ be the line segment between m_1 and m_2 which is folded on w_1. If l_1 is tangent to $p(r, a_1)$ and the reflection of l_1 is tangent to $p(a_1, a_2)$, $m_1m_2^*$ is the tangent to $p(r, a_1)$ and $p(a_1, a_2)$.

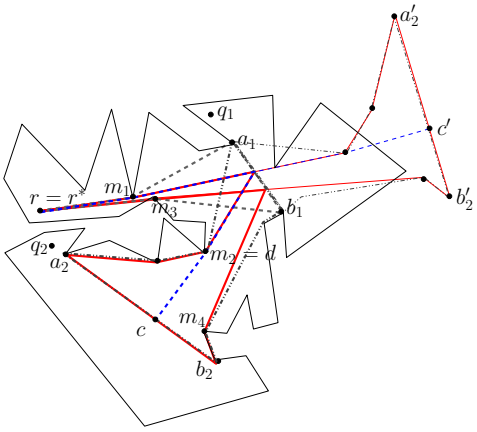


Figure 4: This drawing shows F^* and the optimal point c as well as H_{12}' and the 'unfolded' version of F^*.

Computing the optimal point on the window. We compute the optimal point c on w_2' in F^* in almost the same way as in Section 2.2. Additionally the following issues have to be considered:

- We have to reflect w_2 at w_1, the reflection w_2' is obtained in $O(1)$ time.
- We have to reflect the extension of an edge v_iv_{i+1} in $O(1)$ time on the window w_1 for $v_iv_{i+1} \subseteq p(m_2, a_2)$ or $v_iv_{i+1} \subseteq p(m_4, b_2)$ if we want to compute the angle θ_i.
- If we want to perform a binary search on $p(r, a_2)'$ we first have to look at the angle between w_2' and the extension edges incident to m_2. Therefore we have to reflect m_2 on w_1. Via this angle we can decide if we have to perform a binary search on $p(r, m_1)$ or $p(m_2, a_2)$. Note that, if we perform a binary search on $p(m_2, a_2)$, i.e., on $[m_2, a_2]$, we compute $LCA_{T_{m_2}}(u, v_1)$ for a vertex $u \in [m_2, a_2]$.
- If we want to perform a binary search on $p(r, b_2)'$ we proceed analogously.

In the end, we can compute the length of the shortest path in constant time from the information stored in T_s. E.g., if c is the foot of the perpendicular from $d \in p(m_2, a_2)$, the length of the shortest path is the sum of the lengths of $p(s, m_1)$, $m_1m_2^*$, $p(m_2, d)$ and dc. The shortest path itself can be output in time linear in its length.

4 Conclusion

It remains open if the problem can still be solved efficiently for polygons with holes. It also remains unclear if the algorithm can be generalized to more than two query points.

References

[1] S. Ghosh. *Visibility Algorithms in the Plane.* Cambridge University Press, 2007.

[2] L. J. Guibas, J. Hershberger. *Optimal shortest path queries in a simple polygon.* Journal of Computer and System Sciences, 39(2):126–152, 1989.

[3] L. J. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan. *Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons.* Algorithmica, 2:209–233, 1987.

[4] D. Harel and R. E. Tarjan. *Fast algorithms for finding nearest common ancestors.* SIAM J. Comput., 13(2):338–355, 1984.

[5] J. Hershberger and S. Suri. *A pedestrian approach to ray shooting: Shoot a ray, take a walk.* J. Algorithms, 18:403–431, 1995.

[6] R. Khosravi and M. Ghodsi. *Shortest paths in simple polygons with polygon-meet constraints.* Inf. Process. Lett., 91(4):171–176, 2004.

[7] R. Khosravi and M. Ghodsi. *The fastest way to view a query point in simple polygons.* EWCG 2005, Eindhoven, Netherlands, pages 187–190, 2005.

[8] C. Knauer and G. Rote. *Shortest Inspection-Path Queries in Simple Polygons. Technical report B-05-05*, Dep. of Computer Science, Freie Universität Berlin, 2005.

[9] M.H. Overmars and J. van Leeuwen. *Maintenance of configurations in the plane.* Journal of Computer and System Sciences, 23:166–204, 1981.

[10] L. Schlipf. *Kontrollpfadanfragen in einfachen Polygonen.* Diplomarbeit, Freie Universität Berlin, 2008.

A Search for Medial Axes in Straight Skeletons

Kira Vyatkina*

Abstract

For a simple polygon P, let $S^c(P)$ denote the subgraph of the straight skeleton for P traced out by the convex vertices of the linear wavefront. We show that $S^c(P)$ decomposes into a set of "pruned" medial axes for certain convex polygons closely related to P, and give an optimal algorithm for computation of those polygons.

1 Introduction

The straight skeleton for a simple polygon was first introduced by Aichholzer et al. [1], and has promptly found a number of applications in such areas as surface reconstruction [3], computational origami [6], and many others. Besides, it has served as a basis for another type of skeleton called linear axis [8]. Its advantage over the only previously known skeleton – the medial axis – resides in the fact that all its edges are straight line segments, while the medial axis for a non-convex polygon necessarily contains parabolic edges as well.

Both the straight skeleton and the medial axis can be defined through wavefront propagation. Initially, the wavefront coincides with the given polygon. To obtain the straight skeleton, we let the wavefront edges move inside the polygon at equal speed, thereby remaining parallel to themselves, and keep track of the movement of all the vertices. The underlying process is referred to as the *linear wavefront propagation*. To obtain the medial axis, we apply a *uniform wavefront propagation*, during which all the wavefront points move inside at constant speed. Thus, at time $t > 0$, the uniform wavefront consists of the interior points of the polygon at the distance t from its boundary. In the process, the medial axis is traced out by the convex vertices of the wavefront.

However, the straight skeleton is computationally more expensive than the medial axis: the fastest known deterministic algorithm for its construction requires $O(n^{1+\varepsilon} + n^{8/11+\varepsilon}r^{9/11+\varepsilon})$ time and space [7], where r is the number of reflex vertices of the polygon, and ε is an arbitrarily small positive constant. The best existing randomized algorithm computes the straight skeleton for a non-degenerate simple polygon in $O(n\log^2 n + r\sqrt{r}\log r)$ expected time; for a degenerate one, the expected time bound amounts to $O(n\log^2 n + r^{17/11+\varepsilon})$ [4]. But the medial axis for a simple polygon can be obtained in linear time [5]. It is a common belief that the straight skeleton can be computed in a more efficient way than it is possible nowadays. Yet development of such methods is likely to require investigation of additional properties of the straight skeleton. In this work, we take one step further in that direction.

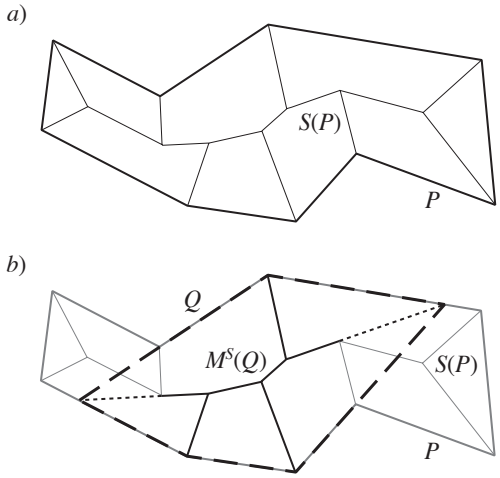


Figure 1: a) A simple polygon P and its straight skeleton $S(P)$. b) The subtree $M^S(Q)$ of $S(P)$ is a pruned medial axis for the convex polygon Q.

Our main observation is that during the linear wavefront propagation, the pieces of the wavefront locally interact exactly in the same way as if they originated from the boundary of a (bounded or unbounded) convex polygon, the sides of which either coincide or overlap with certain sides of the given polygon P. Since for a convex polygon, the two kinds of propagation proceed identically, this implies that some pieces of the medial axes for such polygons are embedded in the straight skeleton $S(P)$ for the polygon P. We formalize our ideas by indicating those pieces in $S(P)$, providing an efficient algorithm for computation of the corresponding convex polygons, and pointing out that for each such polygon Q, the piece $M^S(Q)$ of its medial axis $M(Q)$ present in $S(P)$ can be obtained by appropriately trimming the edges of $M(Q)$ incident to the vertices of Q not being those of P, and the unbounded ones, if any exist (Fig. 1a,b). Therefore, we say that $M^S(Q)$ is a *pruned* medial axis for Q.

*Department of Mathematics and Mechanics, Saint Petersburg State University, kira@meta.math.spbu.ru

2 Event taxonomy

Let P be a simple polygon with n vertices, r of those being reflex; assume that $r \geq 1$. Consider the process of constructing the straight skeleton $S(P)$ for P through linear wavefront propagation. During the propagation, the structure of the wavefront changes in a discrete manner at certain events; thereby, some old vertices vanish, and new ones appear in the wavefront. The basic event taxonomy [1] distinguishes between *edge events*, at which a wavefront edge shrinks to zero, and *split events* that occur when a reflex vertex collides with an edge, causing a split of a connected wavefront component into two. More elaborated classifications also recognize *vertex events*, which correspond to collisions between two or more reflex vertices, with nothing else being at the same place. Vertex events may lead to appearance of new reflex vertices in the wavefront, and are often treated as a special degenerate case [4, 7]. We shall first assume that no vertex events occur during construction of $S(P)$, and give remarks on handling degeneracies at the end of our exposition. In the absence of vertex events, the trace of the reflex vertices is given by the union of the edges of $S(P)$ incident to the reflex vertices of P. Also, under this assumption, any node of $S(P)$ of degree $d \geq 4$ is produced by $(d-2)$ edge and/or split events that simultaneously occur at the same location. Those events can be handled one at a time, with any two consecutive ones being separated by a zero time interval. Therefore, any node of $S(P)$ having degree $d \geq 4$ can be interpreted as $(d-2)$ coinciding nodes of degree three connected by $(d-3)$ edges of zero length in such a way that the subgraph induced by those nodes is a tree. Consequently, we may further suppose that any inner node of $S(P)$ has degree three.

3 Decomposition of the straight skeleton

Let us remove from $S(P)$ all the edges incident to the reflex vertices of P. The remaining subgraph $S^c(P)$ of $S(P)$ has been traced out by the convex vertices of the linear wavefront. Next, split $S^c(P)$ at the inner nodes that were incident to the deleted edges (Fig. 2a,b). As a result, we obtain a decomposition of $S^c(P)$ into $k \leq r+1$ connected components, which we denote by M_1, $M_2, \ldots, M_k$. We claim that for any i, $1 \leq i \leq k$, M_i is a part of the medial axis for a convex polygon. Strictly speaking, there are infinitely many such polygons; of course, we would like to retrieve one with the least computational effort. Below we shall formalize our intent.

For any edge e of P, we define its corresponding *cell* $C(e)$ to be the face of the partition of P induced by $S(P)$, which is adjacent to e. Equivalently, $C(e)$ is the region swept in the propagation by the portion of the linear wavefront originating from e.

Consider any M_i. Since $S(P)$ is a tree, M_i is a tree as well. The embedding of M_i in the plane induces a cyclic order of its leaves. For any consecutive pair of leaves, when walking from one of them to the other along the edges of M_i, we follow the boundary of some cell. Moreover, in can be easily verified that for any two such pairs of leaves, the corresponding cells must be different. These cells are also cyclically ordered, in compliance with the ordering of the leaves.

Now retrieve all the edges of P, such that the boundaries of their cells contribute to M_i; denote the resulting set by E_i (Fig. 2c). From the above discussion, it follows that $|E_i|$ equals the number of the leaves in M_i. Let the edges in E_i inherit the cyclic order of the cells. For any two consecutive edges $e, e' \in E_i$, their cells $C(e)$ and $C(e')$ share an edge of M_i incident to a leaf. If the leaf corresponds to a convex vertex of P, then e and e' share this vertex. Otherwise, the leaf corresponds to an inner node u of $S(P)$ adjacent to a reflex vertex of P. In this case, e and e' can be (but not necessarily are) adjacent only if E_i consists solely of e and e'. To see this, suppose e and e' are adjacent. Then $C(e)$ and $C(e')$ must share an edge of $S(P)$ incident to a vertex of P. On the other side, any two cells can share at most one edge of $S(P)$. Therefore, the edge of M_i shared by $C(e)$ and $C(e')$ must have one endpoint at u, and the other – at a convex vertex of P. Consequently, M_i consists of a single edge, which is incident to two cells $C(e)$ and $C(e')$, and e and e' are the only two edges in E_i.

Thus, we conclude that the edges from E_i together compose one or a few disjoint convex chains cut out of the boundary of P. Let $\mathcal{C}_i = \{c_1^i, \ldots, c_{m_i}^i\}$ denote the set of those chains; observe that m_i equals the number of the leaves of M_i that correspond to the inner nodes of $S(P)$. Denote by $f(c_j^i)$ and $l(c_j^i)$ the first and the last edge of the chain c_j^i, respectively; assume that c_j^i is traversed from $f(c_j^i)$ to $l(c_j^i)$ when walking counterclockwise along the boundary of P, where $1 \leq j \leq m_i$. If c_j^i consists of a single edge, then $f(c_j^i)$=$l(c_j^i)$. Let $v_f(c_j^i)$ and $v_l(c_j^i)$ denote the first and the last vertex of c_j^i, respectively. Without loss of generality, suppose that the order of the chains in $\mathcal{C}_i$ corresponds to the one, in which they are traversed when walking counterclockwise along the boundary of P. To unify the notation, let $c_{m_i+1}^i = c_1^i$, and let $c_0^i = c_{m_i}^i$.

Lemma 1 *For any j, $1 \leq j \leq m_i$, at least one of $v_l(c_j^i)$ and $v_f(c_{j+1}^i)$ is reflex.*

Let us take any chain c_j^i and prolong $l(c_j^i)$ to infinity, thereby eliminating $v_l(c_j^i)$. Denote the resulting chain by $\overrightarrow{c}^i_j$, and its last (unbounded) edge – by $\overrightarrow{l}(c_j^i)$. Similarly, let $\overleftarrow{c}^i_j$ denote the chain obtained from c_j^i by appropriately prolonging $f(c_j^i)$ to infinity,

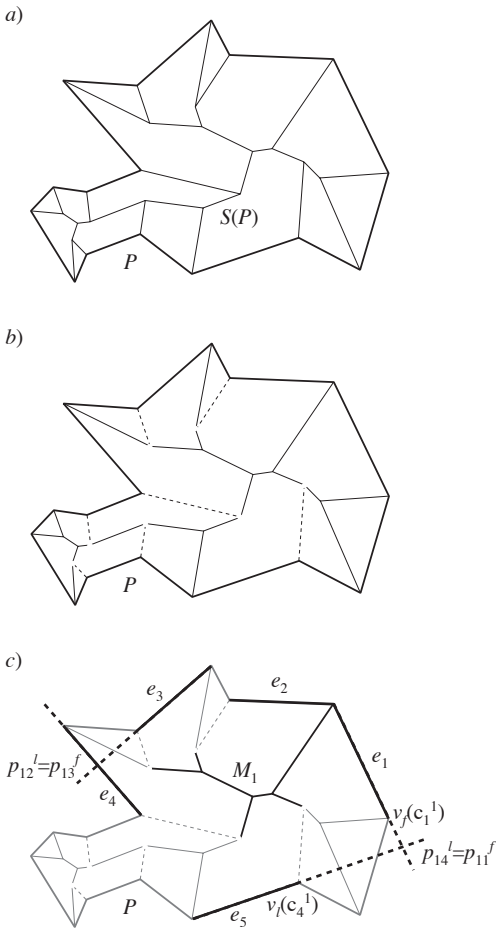


Figure 2: a) A simple polygon P and its straight skeleton $S(P)$. b) Decomposition of the subgraph $S^c(P)$ of $S(P)$ traced out by the convex vertices of the linear wavefront. The edges of $S(P)$ traced out by the reflex vertices are shown dashed. c) For the subtree M_1 of $S(P)$, $E_1 = \{e_1, e_2, e_3, e_4, e_5\}$; $\mathcal{C}_1 = \{c_1^1, c_2^1, c_3^1, c_4^1\}$, where the chain c_1^1 is formed of e_1 and e_2, and each of c_2^1, c_3^1, and c_4^1 consists of a single edge – of e_3, e_4, and e_5, respectively. By prolonging the edges as shown in dotted lines, we obtain the chains $\overleftarrow{c}^1_1$, $\overrightarrow{c}^1_2$, $\overleftarrow{c}^1_3$, and $\overrightarrow{c}^1_4$, respectively; $p^l_{14} = p^f_{11} = \overrightarrow{l}(c^1_4) \cap \overleftarrow{f}(c^1_1)$, and $p^l_{12} = p^f_{13} = \overrightarrow{l}(c^1_2) \cap \overleftarrow{f}(c^1_3)$.

and let $\overleftarrow{f}(c^i_j)$ denote its first (unbounded) edge.

Lemma 2 *For any j, $1 \le j \le m_i$, if $\overrightarrow{c}^i_j \cap \overleftarrow{c}^i_{j+1} \neq \emptyset$, then $\overrightarrow{c}^i_j$ and $\overleftarrow{c}^i_{j+1}$ intersect at a point $p = \overrightarrow{l}(c^i_j) \cap \overleftarrow{f}(c^i_{j+1})$.*

For any j, $1 \le j \le m_i$, if $\overrightarrow{l}(c^i_{j-1})$ and $\overleftarrow{f}(c^i_j)$ intersect, let $p^f_{ij} = \overrightarrow{l}(c^i_{j-1}) \cap \overleftarrow{f}(c^i_j)$; otherwise, let p^f_{ij} be an artificial point at infinity lying on $\overleftarrow{f}(c^i_j)$. Similarly, if $\overrightarrow{l}(c^i_j)$ and $\overleftarrow{f}(c^i_{j+1})$ intersect, let $p^l_{ij} = \overrightarrow{l}(c^i_j) \cap \overleftarrow{f}(c^i_{j+1})$; otherwise, let p^l_{ij} be an artificial point at infinity lying on $\overrightarrow{l}(c^i_j)$ (see Fig. 2c). Note that $p^l_{i,j-1}$ and p^f_{ij} either both are finite and coincide, or both are infinite.

Lemma 3 *Let c^i_j be a chain consisting of a single edge. Then p^f_{ij} lies on the same side of p^l_{ij} as $v_f(c^i_j)$.*

For each j, construct a chain $\overline{c}^i_j$ from c^i_j by adjusting the first and the last edge of the latter, so that they will terminate at p^f_{ij} and p^l_{ij}, respectively. Correctness of the construction is assured by Lemmas 2 and 3. Let $\overline{c}^i = \cup_j \overline{c}^i_j$.

Lemma 4 *$\overline{c}^i$ bounds a convex region in the plane.*

To make it more precise, one of the following three possibilities occurs:

- $\overline{c}^i$ is a closed convex chain, which bounds a convex polygon (Fig. 3a);
- $\overline{c}^i$ is an open convex chain, the first and the last edges of which are infinite, and it bounds an infinite convex region (Fig. 3b);
- $\overline{c}^i$ is formed of two parallel lines, which bound an infinite strip (Fig. 3c).

Let Q_i denote the convex region bounded by the chain $\overline{c}^i$. We shall refer to Q_i as to a convex polygon, either bounded or unbounded. Let $M(Q_i)$ denote the medial axis for Q_i.

Lemma 5 *M_i is part of $M(Q_i)$, and can be obtained from the latter by appropriately trimming its edges incident to the vertices of Q_i not being those of P, and the unbounded ones, if any exist.*

It is easy to see that each M_i is a *maximal* fragment of a medial axis, in a sense that it cannot be extended along the edges of $S(P)$ while remaining a part of the medial axis for any polygon.

Given P and $S(P)$, and assuming that the representation of the latter provides information on the partition of P induced by $S(P)$, it is straightforward to decompose $S^c(P)$ into the subtrees $M_1, \ldots, M_k$, and to retrieve the corresponding sets of chains $\mathcal{C}_1, \ldots, \mathcal{C}_k$. For any i, $1 \le i \le k$, the convex polygon Q_i can then be constructed from $\mathcal{C}_i$ following the procedure described above.

We summarize our results in the next Theorem.

Theorem 6 *Let P be a simple polygon. The part $S^c(P)$ of the straight skeleton $S(P)$ for P, traced out by the convex vertices of the linear wavefront, can be uniquely decomposed into a set of maximal fragments of medial axes. Each of those fragments represents a pruned medial axis for a certain convex polygon. Both the decomposition and the corresponding set of convex polygons can be computed from P and $S(P)$ in linear time.*

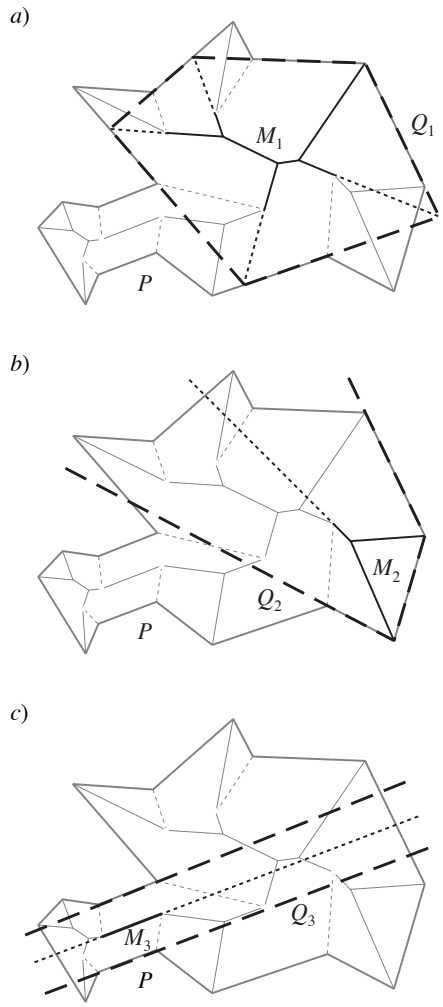


Figure 3: A simple polygon P and its straight skeleton $S(P)$ are depicted gray. The convex region Q_i bounded by $\overline{c}^i$ (dashed) can be of one of the three types: a) a convex polygon; b) an infinite convex region bounded by a chain, the first and the last edges of which are infinite; c) an infinite strip bounded by two parallel lines. For any Q_i, the corresponding subtree M_i of $S(P)$ is shown bold; parts of edges of $M(Q_i)$ not belonging to M_i are shown dotted, where $1 \leq i \leq 3$.

If vertex events do occur during the construction of $S(P)$, the trace of the reflex vertices of the linear wavefront represents a forest F of subtrees of $S(P)$. In particular, all the reflex vertices of P are leaves of F. Removal of F gives us a subgraph $S^c(P)$ of $S(P)$, the edges of which has been traced out by the convex vertices of the linear wavefront. Having split $S^c(P)$ at the inner nodes, at which the reflex wavefront vertices vanish, we obtain a decomposition of $S^c(P)$ into several connected components $M_1, \ldots, M_k$. Following essentially the same considerations as given above, we can show any M_i to be a part of the medial axis for a certain convex polygon, where $1 \leq i \leq k$.

4 Conclusion

For a simple polygon P, we have demonstrated that the subgraph $S^c(P)$ of its straight skeleton $S(P)$, traced out by the convex vertices of the linear wavefront, decomposes into a set of pruned medial axes for certain convex polygons, which can be easily reconstructed from P and $S(P)$ in linear time. Moreover, each element of the proposed decomposition is a maximal fragment of a medial axis, in a sense that it cannot be extended along the edges of $S(P)$ while remaining a part of the medial axis for any polygon.

The concept of the straight skeleton was generalized by Aichholzer and Aurenhammer to the case of planar straight line graphs [2]. So, a natural generalization of our results would be to extend them accordingly. Another direction for future research is to attempt to compute the straight skeleton as a union of the pruned medial axes.

Acknowledgment

This research was supported by Russian Foundation for Basic Research (grant 07-07-00268-a).

References

[1] O. Aichholzer, F. Aurenhammer, D. Alberts, B. Gärtner. *A novel type of skeleton for polygons.* The Journal of Universal Computer Science, 1:752–761, 1995.

[2] O. Aichholzer, F. Aurenhammer. *Straight skeletons for general polygonal figures.* In Proc. 2nd Ann. Int'l. Computing and Combinatorics Conf. COCOON'96, Hong Kong, LNCS 1090, pp. 117–126, 1996.

[3] G. Barequet, M. T. Goodrich, A. Levi-Steiner, D. Steiner. *Contour interpolation by straight skeletons.* Graphical Models (GM) 66(4):245–260, 2004.

[4] S. W. Cheng, A. Vigneron. *Motorcycle graphs and straight skeletons.* Algorithmica 47(2):159–182, 2007.

[5] F. Chin, J. Snoeyink, C. Wang. *Finding the Medial Axis of a Simple Polygon in Linear Time.* Discrete and Computational Geometry 21(3):405–420, 1999.

[6] Demaine, E.D., Demaine, M.L., Lubiw, A.: *Folding and cutting paper.* In Revised Papers from the Japan Conference on Discrete and Computational Geometry (JCDCG'98), LNCS 1763, pp. 104–117, 1998.

[7] D. Eppstein, J. Erickson. *Raising roofs, crashing cycles, and playing pool: applications of a data structure for finding pairwise interactions.* Discrete and Computational Geometry 22(4):569–592, 1999.

[8] M. Tănase, R. C. Veltkamp. *Straight skeleton approximating the medial axis.* In Proc. 12th Annual European Symposium on Algorithms, pp. 809–821, 2004.

On Computing Integral Minimum Link Paths in Simple Polygons

Wei Ding*

Abstract

We consider the problem of finding the minimum link path connecting two points in a simple polygon under the restriction that the interior vertices of this path have integral coordinates. We show that this problem is NP-complete even if the underlying polygon is restricted to be monotone. On the positive side we give a polynomial time 4-approximation algorithm. The underlying model of computational complexity is the bit model with constant cost per bit operation, and not the usual real RAM with constant cost per arithmetic or comparison operation on arbitrary real numbers.

1 Introduction

Connecting two points s and t inside a simple polygon P by polygonal path consisting of as few segments as possible was investigated in the context of computational geometry already in the mid 1980's by researchers such as H. El Gindy, S. Suri, S.K. Ghosh, J. Hershberger, and J. Snoeyink, see [2] for an extensive survey. They showed that such a "minimum link path" could be constructed in time linear in n, the number of vertices of the polygon P. About a decade later S. Kahan and J. Snoeyink [4] pointed out that this result was only true in the real RAM model, and that this computational model was not really appropriate for this type of problem: They exhibited classes of example polygons with vertices drawn from the $N \times N$ integer grid, where the natural coordinate representation of the minimum link path between two points needed $\Omega(n^2\phi)$, $\phi = \log N$ bits. Thus in the bit model of computation any algorithm for explicitly producing a minimum link path needs time and also space at least $\Omega(n^2\phi)$.

Kahan and Snoeyink proposed to improve the complexity by considering only paths whose interior vertices are restricted to lie on some finite point set $R(P)$. They considered two alternatives:

In the first they took $R(P)$ to be the vertices of the arrangement of the $\binom{n}{2}$ lines spanned by the n vertices of P. They showed that this kind of restriction increases the link-distance between two points s and t at most by a factor of 2 over the unrestricted link-distance $\mathrm{ld}_P(s,t)$ between s and t. Moreover such a restricted path could be found in $O(n)$ time (which is to mean with $O(n)$ operations on integers representable by $O(\phi)$ bits).

In the second alternative they took $R(P)$ to be the integer grid points contained in P. They showed that this restriction can increase the link-distance between two points by a multiplicative factor of $\Theta(\phi)$ over the unrestricted link-distance $\mathrm{ld}_P(s,t)$, but not by more. Here it is assumed that the n vertices of P are drawn from the $N \times N$ integer grid. They also gave an algorithm that produced such a grid-restricted link path between points s and t consisting of $O(\mathrm{ld}_P(s,t)\phi)$ with $\phi = \log N$ segments in time $O(n + \mathrm{ld}_P(s,d)\phi)$.

In this note we consider further that second alternative, where the corners of the paths are restricted to integer grid points in P. On the negative side we show that the problem of computing the minimum-link path with this restriction is NP-complete. This holds even if P is monotone. On the positive side we give for x-monotone polygons polynomial time 4-approximation algorithms, one algorithm produces a grid restricted path, the other computes a grid restricted simple path, i.e. it has no self-intersections, connecting s and t that has at most 4 times as many segments than any such restricted path between s and t.

2 Preliminaries and statement of results

Let P be a simple polygon in the plane with n vertices, all from the $N \times N$ integer grid. For $s, t \in P \cap \mathbb{Z}^2$ let *link-path* $\Pi_P(s,t)$ denote the set of all polygonal chains connecting s and t contained in P, let *integer link-path* i-$\Pi_P(s,t)$ be all paths in $\Pi_P(s,t)$ all whose vertices lie on the integer grid, let *simple integer link-path* si-$\Pi_P(s,t)$ denote the set of simple paths in i-$\Pi_P(s,t)$. For path $\pi \in \Pi_P(s,t)$ we define its *link-length* $\mathrm{ll}(\pi)$ to be the number of segments forming π. For $s, t \in P$ define their *link-distance* $\mathrm{ld}_P(s,t)$ as $\min\{\mathrm{ll}(\pi) \mid \pi \in \Pi_P(s,t)\}$, their *integer link-distance* i-$\mathrm{ld}_P(s,t)$ as $\min\{\mathrm{ll}(\pi) \mid \pi \in \text{i-}\Pi_P(s,t)\}$, and their *integer simple link-distance* si-$\mathrm{ld}_P(s,t)$ as $\min\{\mathrm{ll}(\pi) \mid \pi \in \text{si-}\Pi_P(s,t)\}$.

In the rest of the paper we sketch proofs of the following results:

Theorem 1 *The problem of determining whether i-$ld_P(s,t) = k$, given polygon P, integer k, and $s, t \in P$ is NP-complete. This holds true even if P*

*Department of Computer Science, Saarland University, wding@cs.uni-sb.de

is a monotone polygon. The same results hold also for si-ld$_P(s,t)$.

Theorem 2 *Given an x-monotone polygon P with vertices from the $N \times N$ integer grid and two points $s, t \in P$,*

1. *a path $\pi \in i\text{-}\Pi_P(s,t)$ with $ll(\pi) \leq 4 \cdot i\text{-}ld_P(s,t)$ can be found in time $O(n^2 \cdot \phi)$ measured in the bit model of computation.*
2. *a path $\pi \in si\text{-}\Pi_P(s,t)$ with $ll(\pi) \leq 4 \cdot si\text{-}ld_P(s,t)$ can be found in time $O(n^4 \cdot \phi)$.*

3 Integral minimum link integer path is NP-hard

The purpose of this section is to prove Theorem 1. Clearly the problem at hand is in NP. To prove NP-hardness we use a reduction from the well-known NP-complete problem *PARTITION*.

Let $M = \{m_0, m_1, \ldots, m_{n-1}\}$ be an instance of *PARTITION*, and $S = \sum_{m \in M} m$. The purpose of the *PARTITION* problem is to decide, whether there is a subset $M' \subset M$, so that $\sum_{m \in M'} m = \frac{1}{2}S$.

We construct a polygon P_M out of M and points $s, t \in P_M$, such that:

$$\text{i-ld}_P(s,t) = 3n \Leftrightarrow \exists M' \subset M, \sum_{m \in M'} m = \frac{1}{2} \sum_{m \in M} m.$$

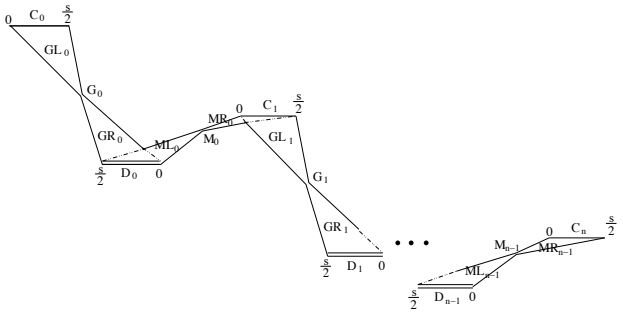


Figure 1: P_M

P_M is a polygon with $2n$ narrow gaps $G_0, M_0, G_1, M_1, \ldots, G_{n-1}, M_{n-1}$ from left to right, as depicted in Figure 1. $C_i, 0 \leq i \leq n$, in the figure is a horizontal line segment that contains $\frac{S}{2} + 1$ integer points representing the number 0 to $\frac{S}{2}$. Let $c_i(v)$ denotes the point in C_i that represents the value v. $D_i, 0 \leq i \leq n-1$, contains $2(\frac{S}{2}+1)$ integer points on two horizontal line segments. Each value $v \in \{0, 1, \ldots, \frac{S}{2}\}$ finds two representations in D_i, $d_i^o(v)$ and $d_i^b(v)$ denote the two integer points of value v on the top and bottom segments in D_i. GL_i is the area between C_i and the gap G_i, GR_i is the area between D_i and G_i, ML_i is the area between D_i and the gap M_i, MR_i is the area between C_{i+1} and M_i.

The gaps are constructed with the following restrictions:

1. i-ld$_P(c_i(v), d_i^o(v)) = 1$; i-ld$_P(c_i(v), d_i^o(u)) > 1$, for $u \neq v$.
2. i-ld$_P(c_i(v), d_i^b(v)) = 1$; i-ld$_P(c_i(v), d_i^b(u)) > 1$, for $u \neq v$.
3. i-ld$_P(d_i^b(v), c_{i+1}(v)) = 2$; i-ld$_P(d_i^b(v), c_{i+1}(u)) > 2$, for $u \neq v$.
4. i-ld$_P(d_i^o(v), c_{i+1}(v + m_i)) = 2$; i-ld$_P(d_i^b(v), c_{i+1}(u)) > 2$, for $u \neq v + m_i$.
5. $GR_i \cap ML_i \cap \mathbb{Z}^2 = D_i \cap \mathbb{Z}^2$.
6. $MR_i \cap GL_{i+1} \cap \mathbb{Z}^2 = C_{i+1} \cap \mathbb{Z}^2$.
7. $s = c_0(0)$; $t = c_n(\frac{S}{2})$.

Items 1 and 2 imply, from the integer point $c_i(v)$ the two points $d_i^o(v)$ and $d_i^b(v)$ in D_i are exclusively visible. 3 and 4 mean that from a value v in D_i, either v or $v + m_i$ is reached in C_i through two segments. Restriction 5 and 6 show that in order to reach an integer point in C_{i+1} from an integer point in C_i with 3 segments, an integer link-path have to turn at an integer point at D_i.

The following holds for the P_M satisfying these 6 restrictions: $\forall v \in \{\sum_{0 \leq j \leq i-1} s_j m_j\}$, $s_j \in \{0,1\}$, i-ld$_{P_M}(s, c_i(v)) = 3i$; $\forall v \notin \{\sum_{0 \leq j \leq i-1} s_j m_j\}$, $s_j \in \{0,1\}$, i-ld$_{P_M}(s, c_i(v)) > 3i$.

It follows that i-ld$_{P_M}(s, c_n(v)) = 3n$ for $v \in \{\sum_{0 \leq j \leq n-1} s_j m_j\}$, which means, for all $M' \subset M$, i-ld$_{P_M}(s, c_n(\sum_{m \in M'} m)) = 3n$. In other words, i-ld$_{P_M}(s,t) = 3n \Leftrightarrow \exists M' \subset M, \sum_{m \in M'} m = \frac{S}{2}$.

In the following part we explain the geometry of the P_M. We call the part between C_i and D_i a *selector*, where an integer path through $c_i(v)$ performs a selection between $d_i^o(v)$ and $d_i^b(v)$. We denote the part between D_i and C_{i+1} an *adder*, an integer path will reach $c_{i+1}(v + m_i)$ from $d_i^o(v)$ and reach $c_{i+1}(v)$ from $d_i^b(v)$.

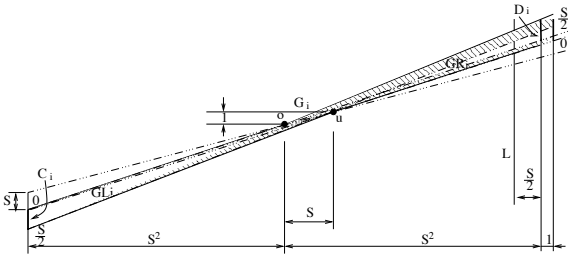


Figure 2: Selector

Figure 2 depicts a selector for space reason rotated by 90^o. The construction achieves, from a point $c_i(v)$, the points $d_i^o(v)$ and $d_i^b(v)$ in D_i are exclusively visible, which guarantees the restrictions 1,2. L is a line that has distance $\frac{S}{2}$ from the component D_i, none integer points between L and D_i is visible from some integer point in C_i, which realizes the restriction 5.

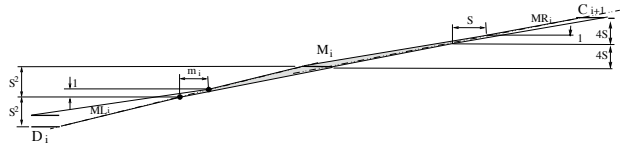


Figure 3: Adder

Figure 3 shows an adder of a number m_i. The geometry of this adder guarantees that: From the point $d_i^b(v)$ the point $c_{i+1}(v)$ in C_{i+1} is exclusively reachable with 2 segments. From the point $d_i^o(v)$ the point $c_{i+1}(v + m_i)$ in C_{i+1} is exclusively reachable with 2 segments, correspond to the restrictions 3,4. No integer points under the C_{i+1} is visible from some integer points in D_i, which makes the restriction 6 true.

4 Approximation algorithms

Given an x-monotone polygon P with vertices from the $N \times N$ integer grid and two points $s, t \in P \cap \mathbb{Z}^2$, we compute a path $\pi \in \text{i-}\Pi_P(s,t)$ with $\text{ll}(\pi) \leq 4 \cdot \text{i-ld}_P(s,t)$ in time $O(n^2 \cdot \phi)$, and a path $\pi' \in \text{si-}\Pi_P(s,t)$ with $\text{ll}(\pi') \leq 4 \cdot \text{si-ld}_P(s,t)$ in time $O(n^4 \cdot \phi)$.

We call two areas A_1 and A_2 inside P *integer connective*, if $\forall p_1 \in A_1$, $\exists p_2 \in A_2$ with $\overline{p_1 p_2} \subset P$; $\forall p_2 \in A_2$, $\exists p_1 \in A_1$ with$\overline{p_1 p_2} \subset P$ and there are a $p_1 \in A_1 \cap \mathbb{Z}^2$, a $p_2 \in A_2 \cap \mathbb{Z}^2$.

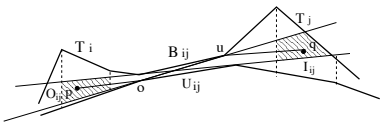


Figure 4: 3-approximation of the minimum link integer path

An x-monotone polygon P of n vertices is bounded by two x-monotone polylines *upper chain* and *lower chain*. W.l.o.g, assume no vertices have the same x-coordinate. P is partitioned into trapezoids $T_1, \ldots, T_{n-1}$ from left to right, see Figure 4. W.l.o.g, we suppose s is the left-most vertex of P and t is the right-most vertex of P, let $T_0 = \{s\}, T_n = \{t\}$. Two trapezoids T_i and T_j are called *integer visible*, denoted as $\text{IV}(T_i, T_j)$, if there exist $A_i \subset T_i$ and $A_j \subset T_j$, so that A_i and A_j are integer connective.

How do you decide, whether two trapezoids T_i and $T_j, j > i$ are integer visible? We consider the parts of the upper chain and the bottom chain between trapezoids T_i and T_j. Let B_{ij} be the bottom convex hull of the upper chain and U_{ij} be the upper convex hull of the bottom chain. If B_{ij} and U_{ij} intersect in more than one segment, then T_j and T_i are not visible. Otherwise we find the two inner tangent of B_{ij} and U_{ij}. These two tangent lines form a wedge O_{ij} in T_i and a wedge I_{ij} in T_j, these two wedges are connective. Furthermore if both O_{ij} and I_{ij} contain integer points, then O_{ij} and I_{ij} are integer connective, and T_i and T_j are integer visible.

Corollary 3 *$IV(T_i, T_j) \Leftrightarrow \forall p \in O_{ij} \cap \mathbb{Z}^2 \wedge q \in I_{ij} \cap \mathbb{Z}^2$, i-ld$_P(pq) \leq 3$.*

Proof. For integer points $p \in O_{ij}$ and $q \in I_{ij}$, $\pi \in \text{i-}\Pi_P(p,q)$ is a path through p, o, u, q, where o, u are a pair of inner tangent points of the convex hulls B_{ij} and U_{ij}, see Figure 4. □

4.1 Computing an integer link-path

To compute an integer link-path in P between $s, t \in P \cap \mathbb{Z}^2$, we at first construct a *visible graph* G_P, then we do a breadth first search on G_P.

G_P is defined as $G_P = (V, E)$, with $V = \{T_0, T_1, \ldots, T_n\}$, $E = \{(T_i, T_j) | IV(T_i, T_j)\}$. G_P can be computed in time $O(n^2) \cdot \phi$. In n^2 time we can find wedges O_{ij} and I_{ij} for all pairs of trapezoids T_i, T_j, see [1], to check if O_{ij} and I_{ij} contain integer points need time $O(\phi)$ according to [3].

Do a breadth first search on G_P, starting from the node T_0, until the node t_n is found. T_0 is at the depth 0 in the breadth first search tree.

Lemma 4 *t_{n+1} is found by a breadth first search in the depth d, then i-ld$_P(s,t) \geq d$ and $d \leq 4 \cdot$ i-ld$_P(s,t)$.*

Proof. There cannot be an integer link-path with link-length less than d. If there exists such a path π of length $l < d$, the vertices of this path are sequentially from the trapezoids $T_0, T_{p1}, T_{p2}, \ldots, T_{p(l-1)}, T_n$, each neighbor of the trapezoids in this list are integer visible to each other, then the breadth first search will stop at the depth l instead of d.

Suppose $T_0, T_{p1}, T_{p2}, \ldots, T_{p(l-1)}, T_n$ is the path form T_0 to T_n in the breadth first search tree. Connect each pair of successive trapezoids in the list with an integer Link-path of link-length 3, see Corollary 3. Call the path between T_{pi} and $T_{p(i+1)}$ *subpath* $SP(T_{pi}, T_{p(i+1)})$. Connect sequentially these subpaths to form an integer link path from s to t, this path is a path π with $\text{ll}(\pi) \leq 4 \cdot \text{i-ld}_P(s,t)$. □

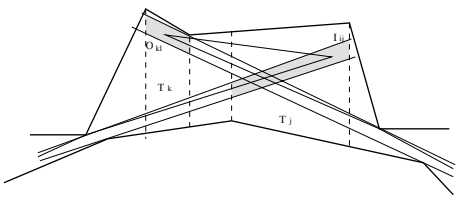


Figure 5: Non-simple path

Note that the path π need not be simple, as show in Figure 5.

4.2 Computing a simple integer link-path

Let's study the visible graph G_P. With breadth first search on G_P we get a list of trapezoids $(T_0, T_{p1}, T_{p2}, \ldots, T_{p(d-1)}, T_n)$, an integer link-path $\pi \in$ i-Π_P is the connection of $SP(T_{pi}, T_{p(i+1)})$'s. As we ask the integer link-path to be simple, a question arises: does $SP(T_{p(i-1)}, T_{pi})$ intersects with $SP(T_{pi}, T_{p(i+1)})$ or other subpaths? We call two edges (t_i, t_j) and (t_l, t_k) in G_P *coincident*, denoted as $coin((t_i, t_j), (t_l, t_k))$

- in case of $i < j = l < k$, see Figure 6 (A), if the shaded area contains at least one integer point.
- in case of $i < l < j < k$, see Figure 6 (B), if the shaded area to the left of segment $\overline{at}$ contains at least one integer point. a is one of the tangent point that forms O_{lk}, t is an integer point lying at the convex hull of integer points in O_{lk} that tangent to a.

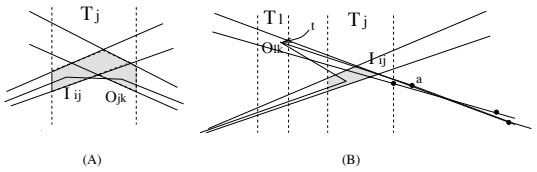


Figure 6: Simple path

Corollary 5 *(t_i, t_j) and (t_l, t_k) are coincident, then for a pair of integer points $p \in O_{ij} \cap \mathbb{Z}^2$ and $q \in I_{lk} \cap \mathbb{Z}^2$, there is a $\pi \in$ si-Π_P, $ll(\pi) \leq 7$.*

Construct a coincidence graph $K_P = (\Upsilon, \Sigma)$, with $\Upsilon = \{v_{i,j} | (T_i, T_j) \in G_P\} \cup (T_0, T_0) \cup (T_n, T_n)$, $\Sigma = \{(v_{i,j}, v_{l,k}) | coin((T_i, T_j), (T_l, T_k))\}$. K_P is constructed in time $O(n^4 \cdot \phi)$. There are n^4 edges in K_P, one decide an edge of case (A) through finding integer points in the shaded area in $O(\phi)$ time, see [3]; in case (B), one find the segment $\overline{at}$ in time $O(\phi)$ using the idea from [3], and find integer points in the shaded area in time $O(\phi)$.

A path L in K_P with $L = (v_{0,0}, v_{i(1),j(1)}, v_{i(2),j(2)}, \ldots, v_{n,n})$ is a *valid* path, if $i(m) > j(m-2), \forall 2 < m < n$. The length of a valid path L is the number of edges in L, denoted as $\text{len}(L)$. Let $\text{vp}(K_P)$ denote the set of all valid paths connecting $v_{0,0}$ and $v_{n,n}$ in K_P. Let $d = min\{l | l = \text{len}(L), L \in \text{vp}(K_P)\}$.

A invalid path has at some place $i(m) \leq j(m-2)$, if $coin((T_{i(m-2)}, T_{j(m-2)}), (T_{i(m)}, T_{j(m)}))$ then we can go directly from node $v_{i(m-2),j(m-2)}$ to $v_{i(m),j(m)}$; the worse case is $coin((T_{i(m-2)}, T_{j(m-2)}), (T_{i(m)}, T_{j(m)}))$ does not hold, one cannot find a $\pi \in$ si-Π_P according to the L.

Lemma 6 *si-$ld_P(s,t) \geq d$; we can find a $\pi \in$ si-Π_P with $ll(\pi) \leq 4 \cdot$ si-$ld_P(s,t)$.*

Proof. Assume there is an path $\pi \in$ si-Π_P with $\text{ll}(\pi) = d' < d$. The vertices of π are sequentially from the trapezoids $T_0, T_{p1}, T_{p2}, \ldots, T_{p(d'-1)}, T_n$, there must be a valid path in K_P in form of $v_{0,0}, v_{o,p1}, \ldots, v_{p(d'-1),n}$, whose length less equal to d'.

If $L = (v_{0,0}, v_{i(1),j(1)}, v_{i(2),j(2)}, \ldots, v_{n,n})$ is the valid path with minimal length, there exists a subpath $SP(T_{i(m)}, T_{j(m)})$ for each $v_{i(m),j(m)} \in L$ such that these subpaths donot intersect each other. Each sunpath has 3 segments and to connect pairs of them we need $d-1$ segments. □

The whole procedure of finding a simple integer link-path takes time $O(n^4 \cdot \phi)$. To initialize G_P and K_P need time $O(n^4 \cdot \phi)$. A path $\pi \in$ si-$\Pi_P(s,t)$ with $\text{ll}(\pi) \leq 4 \cdot$ si-$\text{ld}_P(s,t)$ can be found in time in $O(n^4)$ time.

References

[1] M.Berg and M.Kreveld and Stefan Schirra. A new approach to subdivision simplification. *Twelfth International Symposium on Computer-Assisted Cartography*,volume 4, page 79-88, Charlotte, North Carolina, 1995.

[2] A. Maheshwari, J.-R. Sack and H. Djidjev. Link Distance Problems. *Handbook of Computational Geometry.*

[3] Friedrich Eisenbrand and Günter Rote. Fast 2-Variable Integer Programming. *IPCO*, page 78-89, 1995.

[4] Simon Kahan and Jack Snoeyink. On the Bit Complexity of Minimum Link Paths: Superquadratic Algorithms for Problems Solvable in Linear Time. *Symposium on Computational Geometry*, page 151-158, 1996.

Constant-Working-Space Image Scan with a Given Angle

Tetsuo Asano *

Abstract

This paper proves that there is a linear-time algorithm for scanning an image with a given angle using only constant size of working space. This is an extension of a popular raster scan using $O(1)$ working space.

1 Introduction

There are increasing demands for highly intelligent peripherals such as printers, scanners, and digital cameras. To achieve intelligence they need sophisticated built-in or embedded softwares. One big difference from ordinary software in computers is little allowance of working space which can be used by the software. In this paper we propose a space-efficient algorithm used for image processing. Especially, we present an algorithm for scanning an image with any angle. It is rather easy to do it using only constant-size working space with some sacrifice of running time. This paper achieves the goal without any sacrifice of running time. That is, we present an algorithm for scanning an image consisting of n^2 pixels with an arbitrarily given angle using only constant-size working space which runs in $O(n^2)$ time. More important for embedded software is simpleness of an algorithm. In fact, it requires no sophisticated data structure or recursive function.

The work in this paper would open a great number of possibilities in applications to computer vision, computer graphics, and build-in software design. Image scan is just one of the most important topics in the direction.

2 Rotated raster scan

Let G be an image consisting of $n \times n$ pixels where each pixel has integer coordinates ranging from 0 to $n-1$. It is easy to enumerate all those pixels in a raster order, that is, in the order of $(0,0), (1,0), \ldots, (n-1,0), (0,1), (1,1), \ldots, (n-1,n-1)$. That is,

$$G = \{(x,y) \mid x,y = 0,1,\ldots,n-1\}. \qquad (1)$$

This enumeration can be implemented by a program of double loops:

*School of Information Science, JAIST, Japan, t-asano@jaist.ac.jp

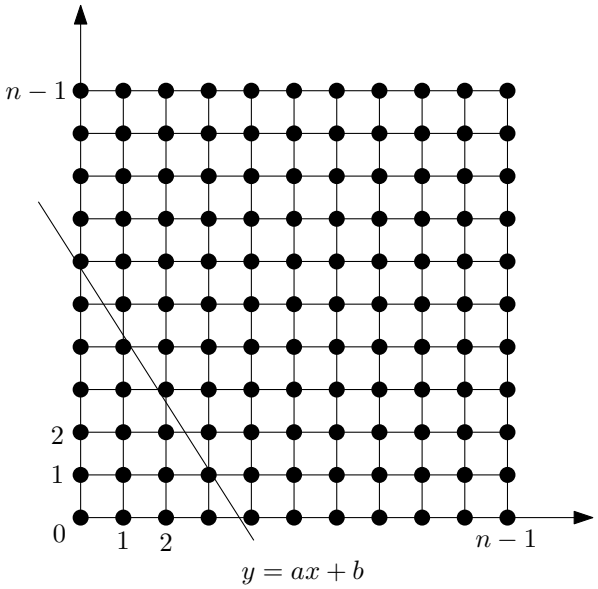


Figure 1: A square image and a line determining scan order.

```
for y = 0 to n-1 do
    for x = 0 to n-1 do
        output a pixel (x, y);
```

This scan is referred to as a row-major raster scan. In a similar way we can define a column-major raster scan. As an extension or generalization we consider a rotated raster scan in which pixels are enumerated along lines of a given angle. Throughout the paper we assume that the angle of those lines is given as a slope a instead of angle and we call the line $y = ax$ the **guide line** for the rotated scan. Figure 1 shows an example of an image and a guide line specifying scan order.

Throughout the paper we assume that the slope a of the guide line is negative to simplify the argument. The case of a positive slope is symmetric. We start from the pixel $(0,0)$. The next point is either $(0,1)$ or $(1,0)$ depending on which is closer to the line $y = ax$ of slope a passing through the origin $(0,0)$. In this way we output pixels in the increasing order of the vertical distances to the line $y = ax$. Therefore, we can describe the rotated raster scan using a priority queue PQ.

Rotated Raster Scan with Slope a
— Algorithm 1:

PQ: priority queue keeping pixels with vertical distances to the line $y = ax$ as keys.
for $x = 0$ to $n - 1$
  Put a pixel $(x, 0)$ into PQ with key $= -ax$.
repeat{
  Extract a pixel (x, y) of the smallest key from PQ.
  Output the pixel (x, y).
  if $(x, y) = (n - 1, n - 1)$ then exit from the loop.
  if $(y < n - 1)$ then put a pixel $(x, y + 1)$ into PQ with key $= y + 1 - ax$.
}

It is easy to see that the algorithm is correct and runs in $O(n^2 \log n)$ time using $O(n)$ working space. Correctness of the algorithm is based on the observation that in each column (of the same x coordinate) pixels are enumerated from bottom to top one at a time. Thus, whenever we output a pixel (x, y), we remove the pixel from the priority queue and insert the pixel just above it, i.e., $(x, y+1)$ if it is still in the image area G.

3 Scanning by rational slope

Recall that the objective of this paper is to reduce the space requirement by an algorithm for scanning an image with a given angle. As a first step we show it is possible to reduce the working space if a slope a of a guide line is a rational number $-q/p$ such that its numerator and denominator are both less than n, the image size, and the two integers p and q are prime to each other.

Now, we want to sort all the pixels in the increasing order of their vertical distances to the guide line $y = ax = -(q/p)x$. If we denote the vertical distance from a pixel (x, y) to the guide line by $d(x, y)$, then we have

$$d(x, y) = y + \frac{q}{p}x = (py + qx)/p. \qquad (2)$$

Since p is fixed, sorting pixels by the values of $d(x, y)$ is equivalent to sorting them by the values $py + qx$.

Now, we partition a set of pixels by lines passing through the bottom pixels $(0, 0), (1, 0), \ldots$. That is, by P_k we denote a set of pixels which lie above the line $y = (-q/p)(x - k)$ and below $y = (-q/p)(x - k - 1)$, or more formally we define

$$\begin{aligned} P_k &= \{(x, y) \in G \mid \\ &\qquad (-q/p)(x - k) \le y < (-q/p)(x - k - 1)\}, \\ k &= 0, 1, \ldots, r,\ r = \lfloor (n - 1)(1 + p/q) \rfloor. \end{aligned}$$

An easy observation here is that if we can order all pixels in each such set then it suffices to output pixels in the order of $P_0, P_1, \ldots, P_r$. Then, how can we order those pixels in a set P_k? We further partition the set of pixels in P_k by their y coordinates, that is, we define a set

$$\begin{aligned} P_k[m] &= \{(x, y) \in P_k \mid m \le y/q < m + 1\}, \\ &\qquad m = 0, 1, \ldots, \lceil n/q \rceil\}. \end{aligned}$$

Figure 2 shows $P_6[0]$ and $P_6[1]$ in an image of size 11×11, where the guide line is $y = (-5/2)x$, i.e., $p = 2, q = 5$, and $n = 11$. The set $P_6[0]$ consists of $(6, 0), (6, 1), (6, 2), (5, 3), (5, 4)$ and $P_6[1]$ consists of $(4, 5), (4, 6), (4, 7), (3, 8), (3, 9)$. Because of the definition, if (x, y) is an element of $P_k[m]$ then $(x-p, y+q)$ is an element of $P_k[m + 1]$ if $(x - p, y + q) \in G$. In the example of Figure 2, $(6, 0) \in P_6[0]$ implies $(6 - 2, 0 + 5) = (4, 5) \in P_6[1]$. More generally, if (x, y) is an element of the set $P_k[0]$ then the pixel $(x - jp, y + jq)$ belongs to the set $P_k[j]$ for any integer j if the pixel is within the image area, that is, if $0 \le x - jp < n$ and $0 \le y + jq < n$. Now we define a natural equivalence relation:

$$\begin{aligned} &(x, y) \equiv (x', y') \Longleftrightarrow (x, y), (x', y') \in G \text{ and} \\ &x \equiv x' \mod p \text{ and } y \equiv y' \mod q. \end{aligned}$$

Two equivalent pixels have the same vertical distance to the guide line. In fact, for two equivalent pixels (x, y) and $(x - jp, y + jq)$ we have

$$\begin{aligned} d(x, y) &= (py + qx)/p = (p(y + jq) + q(x - jp))/p \\ &= d(x - jp, y + jq). \end{aligned}$$

This implies that once we can order elements of the set $P_k[0]$ then it is easy to order all the elements of the set P_k. Whenever we output a pixel (x, y) in $P_k[0]$, we should output its equivalent pixels in $P_k[1], P_k[2], \cdots$ in order before reporting the next pixel in $P_k[0]$.

Now consider how to order pixels in the set $P_k[0]$. Note that the x coordinates of those pixels are between k and $k - p + 1$. If there are two or more pixels in the same x coordinates then they are reported in the increasing y order. Thus, if a priority queue of size p is available, the previous mechanism can order the pixels in the set $P_k[0]$. Thus, we have the following lemma.

Lemma 1 *Given an $n \times n$ image G and a rational slope $a = -q/p$, there is an algorithm for enumerating all pixels in the order determined by the slope which runs in $O(n^2 \log p)$ time using $O(p)$ working space.*

Proof. The x-coordinates of the pixels in the set $P_k[0]$ range from k to $k - p + 1$ (the pixel $(k - p, q)$ belongs to $P_k[1]$). In each column j, the first pixel to be enumerated is the one just above the line $y = (-q/p)(x-k)$ passing through the pixel $(k, 0) \in P_k[0]$. Since the pixels in the columns $k - 1, k - 2, \cdots$ are $(k-1, \lceil q/p \rceil), (k-2, \lceil 2q/p \rceil), \cdots$. In general, the pixel at column $k - j$ is $(k - j, \lceil jq/p \rceil)$. Starting from

those pixels, we repeatedly choose pixels in the increasing order of their vertical distance to the line $y = (-q/p)(x-k)$, that is, $\lceil jq/p \rceil + (q/p)(k-j-k) = \lceil jq/p \rceil - (jq/p)$ since p and q are prime to each other. Whenever we choose a pixel (x, y) then we remove it from the priority queue and then insert a pixel just above it, that is, $(x, y+1)$ as far as it also belongs to the same set $P_k[0]$. Then, we report the pixel (x, y) and then all of its equivalent pixels in order.

The algorithm is as follows:

Rotated Raster Scan with Slope a
— Algorithm 2:

PQ: priority queue keeping pixels with vertical distances to the line $y = ax$ as keys.
for $k = 0$ to $n-1$
  Enumerate all pixels in the set P_k in order.
Enumeration of pixels in P_k**.**
  for $j = 0$ to $p-1$
    put a pixel $(k-j, \lceil jq/p \rceil)$ into PQ with key $= \lceil jq/p \rceil - (jq/p)$.
  repeat{
    Extract a pixel (x, y) having the smallest key out of PQ.
    do{
      Output the pixel (x, y).
      $x = x - p, y = y + q$.
    } while$((x, y) \in G)$
    if $y < (-q/p)(x-k-1)$
      then put a pixel $(x, y+1)$ with key $= y + 1 + (q/p)(x-k)$.
  }

□

Of course, if $p = O(n)$ then this algorithm is not an improvement of the previous trivial algorithm which runs in $O(n^2 \log n)$ time using $O(n)$ working space. Now, a natural question is whether there is an algorithm which runs in linear time using only constant size of space. In the next section we present one such algorithm.

4 Linear-time constant-working-space algorithm

Our goal here is to achieve constant size of working space while keeping the running time linear in the number of pixels. If the goal is just to achieve the constant size working space, then it is rather easy. Let (x_i, y_i) be the current pixel (its initial value is of course $(0, 0)$). To find the next pixel, for each column we compute the pixel just above the line $y = a(x - x_i) + y_i$ and measure the vertical distance from the pixel to the line. We just choose the pixel of the smallest vertical distance. If there is a tie, we prefer one of a smaller x coordinate. Since it is just minimum-finding process, working space we need is just constant size. Of course, for each pixel we need $O(n)$ time to choose the next pixel, and thus $O(n^3)$ time in total.

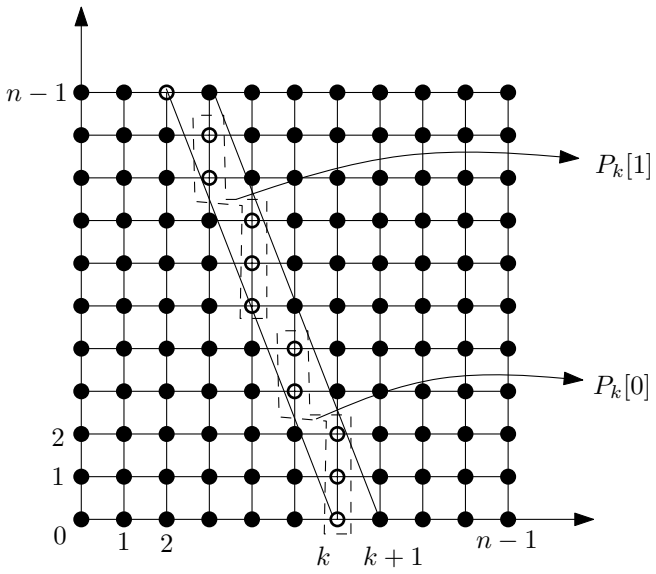


Figure 2: An image with a guide line $y = (-5/2)x$. Two sets of pixels, $P_6[0]$ and $P_6[1]$ are depicted by empty circles.

Our question is whether we can implement the scan using constant-size working space without sacrificing the running time. Fortunately, the answer is yes. A key idea is to use integer properties. The algorithm itself is quite similar to Algorithm 2 described before. An important difference is that no priority queue is used to reduce the working space.

A key is how to order pixels in the set $P_k[0]$. It is known that the first pixel is $(k, 0)$. How can we know the next pixel without using a priority queue? As we have observed before, scan order coincides with the increasing order of the key values $py + qx$ in the set $P_k[0]$. The key value for the starting point $(k, 0)$ is qk. We want to know where is a pixel of key value $qk + i$ for $i = 1, 2, \ldots, p-1$ (note that $P_k[0]$ consists of p pixels).

Suppose we are at a pixel (x, y). If we move to the left, that is, from (x, y) to $(x-1, y)$, then the key decreases by q, and when we move to $(x, y+1)$ then it increases by p. As far as we stay in the same column (x-coordinate), the value of key modulo p remains unchanged. To move to a pixel whose key value is different from the current key value by $i, 0 < i < p$, we have to move horizontally by d_i or $-(p - d_i)$, that is, to $(x + d_i, y)$ or $(x - (p - d_i), y)$, determined by

$$d_i q \equiv i \mod p. \tag{3}$$

Since either $(x+d_i, y)$ or $(x-(p-d_i), y)$ lies in the set $P_k[0]$, we can easily choose the right one. In practice, we do not need all of $d_1, \ldots, d_{p-1}$ since we want to

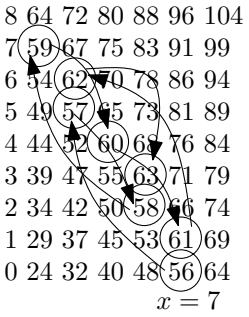


Figure 3: Ordering pixels in the set $P_7[0]$.

find pixels of key values $kq, kq+1, \ldots, kq+p-1$ in an incremental fashion. So, it suffices to know the integer d_1 defined by

$$d_1 q \equiv 1 \mod p. \tag{4}$$

Once we know the x-coordinate and key value of the next pixel, it is easy to calculate its y-coordinate since we know that the key value is given by $py + qx$.

For example, when $p = 5, q = 8$, we have $d_1 = 2$ since $2q \mod p = 16 \mod 5 = 1$. When $k = 7$, the pixel $(7, 0)$ has a key 56. The next pixel is either at $x = 7 + 2 = 9$ or $x = 7 - (5 - 2) = 4$. Since $x = 9$ is outside $P_7[0]$, we can conclude that the next pixel must be at $x = 4$. Now we know the key value is 57. Thus, the y-coordinate is $(57 - 8 \times 4)/5 = 5$. In this way, we can find the next pixel without using a priority queue. See Figure 3.

The algorithm is as follows:

Rotated Raster Scan with Slope a
— Algorithm 3:

Calculate a smallest positive integer d satisfying
  $dq \equiv 1 \mod p$.
for $k = 0$ to $n - 1$
  Enumerate all pixels in the set P_k in order.
Enumeration of pixels in P_k{
  report$(k, 0)$.
  $(x, y) = (k, 0)$.
  for $i = 1$ to $p - 1$ {
    if $x + d \leq k$ then $x' = x + d$
    else $x' = x - (p - d)$.
    $y' = (kq + i)/p$.
    report(x', y').
    $(x, y) = (x', y')$.
  }
}
procedure report(x, y**)**{
  do{
    Output the pixel (x, y).
    $x = x - p, y = y + q$.
  } while$((x, y) \in G)$
}

Algorithm 3 needs a small modification to handle the part after the bottom right corner pixel $(n-1, 0)$, but it is omitted due to page limit.

Theorem 2 *Given an $n \times n$ image G and a rational slope $a = -q/p$, Algorithm 3 above correctly enumerates all the pixels in the order determined by the slope and runs in $O(n^2)$ time using $O(1)$ working space.*

5 Irrational angles

So far we have considered the case where a slope is given as a rational number with its numerator and denominator less than image size. What happens if the slope is irrational? Fortunately, it is not so hard to adapt the proposed algorithm for the case. An important observation is that our guide line is well characterized by which two pixels should be separated. Thus, if we consider a finer grid defined by half grid gap, then such a line is characterized by a line passing through two grid points in the finer grid. Thus, it is not so hard to adapt our algorithm for the case.

6 Applications

Such a rotated scan is required for scanner technology. When we scan a document by a scanner, the document is sometimes rotated. Correction of such rotation is usually done in a scanner. Since scanners cannot possess much working space, space-efficient algorithms are desired. Asano et al. [1] propose a space-efficient algorithm for correcting such a rotation. Especially, it is an in-place algorithm which uses no extra working space other than a given image. In the algorithm a row-major or column-major raster scan is used to scan a rotated subimage. However, by our experience a rotated raster scan sensitive to rotation angle is more desirable.

Acknowledgments

This work of T.A. was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research on Priority Areas and Scientific Research (B). The author would like to express his sincere thanks to Erik Demaine, Stefan Langerman, Ryuhei Uehara, Mitsuo Motoki, and Masashi Kiyomi for their stimulating discussions.

References

[1] T. Asano, S. Bitou, M. Motoki and N. Usui: "In-Place Algorithm for Image Rotation", Proc. ISAAC 2007, pp. 704-715, Sendai, Dec. 2007.

Consistent Digital Rays

Jinhee Chun* Matias Korman* Martin Nöllenburg† Takeshi Tokuyama*

Abstract

Given a fixed origin o in the d-dimensional grid, we give a novel definition of *digital rays* $\mathrm{dig}(op)$ from o to each grid point p. Each digital ray $\mathrm{dig}(op)$ approximates the Euclidean line segment $\overline{op}$ between o and p. The set of all digital rays satisfies a set of axioms analogous to the Euclidean axioms. We measure the approximation quality by the maximum Hausdorff distance between a digital ray and its Euclidean counterpart and establish an asymptotically tight $\Theta(\log n)$ bound in the $n\times n$ grid. Without a monotonicity property for digital rays the bound is improved to $O(1)$.

1 Introduction

The digital line segment $\mathrm{dig}(pq)$ between two grid points p and q is a fundamental digital geometric object, but its definition is not that obvious. Indeed, the digital representation of line segments has been an active subject of research for almost half a century now (see an excellent survey of Klette and Rosenfeld [2]). In digital geometry, a geometric object is represented by a set of d-dimensional grid points in a digital grid $\mathbf{G} = \mathbb{Z}^d$ and its topological properties are considered under a grid topology defined by a graph on the grid. In two dimensions, it is common to consider the orthogonal grid topology, where each point p is connected to the four grid points that are horizontally and vertically adjacent to p, and we focus on this topology; however, as a variant, we may consider the octagonal grid topology that connects each grid point to the eight neighboring grid points with a coordinate difference of at most 1 in each coordinate.

Since a digital line segment is the analogue of a line segment in Euclidean geometry, it is natural to set up the following axioms for a digital line segment:

(S1) A digital line segment $\mathrm{dig}(pq)$ is a connected path between p and q.
(S2) For any two grid points p and q there is a unique digital line segment $\mathrm{dig}(pq) = \mathrm{dig}(qp)$.
(S3) If $s, t \in \mathrm{dig}(pq)$, then $\mathrm{dig}(st) \subseteq \mathrm{dig}(pq)$.
(S4) For any p and q there is a grid point $r \notin \mathrm{dig}(pq)$ such that $\mathrm{dig}(pq) \subset \mathrm{dig}(pr)$.

*GSIS, Tohoku University, Japan. (jinhee, mati, tokuyama)@dais.is.tohoku.ac.jp

†Faculty of Computer Science, Karlsruhe University, Germany. noellenburg@iti.uka.de. Supported by grant WO 758/4-3 of the German Research Foundation (DFG).

Note that axiom (S3) implies that a non-empty intersection of two digital line segments is either a grid point or a digital line segment. Axiom (S4) implies that a digital line segment can be extended to a digital line. We often identify a path in a grid with its vertex set if the correspondence is clear. Accordingly, if we say a grid point p is in a path P, it means that p is a vertex of P.

Unfortunately, popular definitions of two-dimensional (2D) digital line segments in computer vision do not satisfy these axioms. For example, in the standard definition of a *digital straight segment (DSS)* [2], a digital line segment (in the octagonal topology) that corresponds to the line segment $y = mx+b$, $x_0 \le x \le x_1$ is defined as the set of grid points $\{(i, \lfloor mi + b \rfloor) \mid i \in \mathbb{Z},\ x_0 \le i \le x_1\}$ if $|m| \le 1$. Using this definition the intersection of two DSSs is not always connected, and axiom (S3) is violated in some cases.

Another possibility to define digital line segments would be to use the system of L- and Γ-shaped shortest paths. An L- or Γ-shaped path between two points $p = (x_p, y_p)$ and $q = (x_q, y_q)$ such that $x_p \le x_q$, is the (at most) 2-link path that consists of the grid points on the vertical segment pp' and on the horizontal segment $p'q$ where $p' = (x_p, y_q)$. We can confirm that the system of these paths satisfies axioms (S1)–(S4) for digital line segments. A clear drawback is that an L-shaped path is visually very different from the Euclidean line segment, and the Hausdorff distance from $\overline{pq}$ to the L-shaped path becomes $n/\sqrt{2}$ for $p = (0, n)$ and $q = (n, 0)$. Therefore, it seems that there is a trade-off between the axiomatic requirements and the visual quality of digital line segments. It is a challenging problem to find a system of digital line segments that satisfies the axioms and is visually alike Euclidean line segments at the same time.

In this paper we study a less ambitious but important subproblem, motivated by geometric optimization applications like extracting digital star-shaped regions in pixel images [1]: we consider only digital line segments that have a fixed origin o as one of their endpoints. In other words, we consider digital halflines emanating from o, and $\mathrm{dig}(op)$ is defined as the unique portion of the halfline that has p as its second endpoint. We call such segments *digital ray segments* or simply *digital rays* emanating from o.

For digital rays, the axioms for digital line segments should be modified as follows:

(R1) A digital ray dig(op) is a connected path between o and p.
(R2) There is a unique digital ray dig(op) between o and any grid point p.
(R3) If $r \in \mathrm{dig}(op)$, then $\mathrm{dig}(or) \subseteq \mathrm{dig}(op)$.
(R4) For any dig(op), there is a grid point $r \notin \mathrm{dig}(op)$ such that $\mathrm{dig}(op) \subset \mathrm{dig}(or)$.

We also give an additional *monotonicity* axiom, which is not combinatorial, but a reasonable condition for a digital ray:

(R5) For any $r \in \mathrm{dig}(op)$, $|\overline{or}| \leq |\overline{op}|$, where $|\overline{ab}|$ is the length of the Euclidean segment $\overline{ab}$.

A system of digital rays is called *consistent* if it satisfies axioms (R1)–(R5). From these axioms, it follows that the union of all digital rays forms an infinite spanning tree T of the grid graph on $\mathbf{G}$ rooted at o, such that dig(op) is the unique path between o and p in the tree. Because of axiom (R4), T cannot have leaves. Thus, the problem is basically to embed the infinite "star" consisting of the halflines emanating from o in the d-dimensional Euclidean space as a tree in the d-dimensional grid. Although embedding a tree in a grid is a popular topic in metric embedding and graph drawing, it is a novel and interesting problem to geometrically approximate ray segments by paths.

We give the asymptotically tight $\Theta(\log n)$ bound for the maximum Hausdorff distance between dig(op) and $\overline{op}$ among all p in an $n \times n$ grid. The lower bound argument is based on discrepancy theory, and the upper bound is attained by a simple and systematic construction of a tree T that can be extended to the d-dimensional case. Surprisingly, if we do not include the monotonicity axiom (R5), the bound can be reduced to $O(1)$.

2 The lower bound result

The Hausdorff distance $H(A, B)$ of two objects A and B in d-dimensional space is defined by $H(A, B) = \max\{h(A, B), h(B, A)\}$, where $h(A, B) = \max_{a \in A} \min_{b \in B} d(a, b)$ and $d(a, b)$ is some distance between the points a and b. We will use the L_∞-metric in the following for technical convenience, since the choice of the metric is irrelevant if we consider the bounds in big-O and big-Ω notations.

Let's consider the set $V = \{(i, j) \mid i, j \in \mathbb{Z}\}$ of grid points. We define a planar graph G on V that represents the adjacency relations of a pixel grid. In $G = (V, E)$ each vertex (i, j) is connected to its four neighbors $(i, j-1)$, $(i-1, j)$, $(i+1, j)$, and $(i, j+1)$. This also defines the orthogonal topology of the grid $\mathbf{G}$. A subset of V is *connected* in this topology if its induced subgraph in G is connected. We focus on the part $\mathbf{G}(n)$ of the planar orthogonal grid clipped

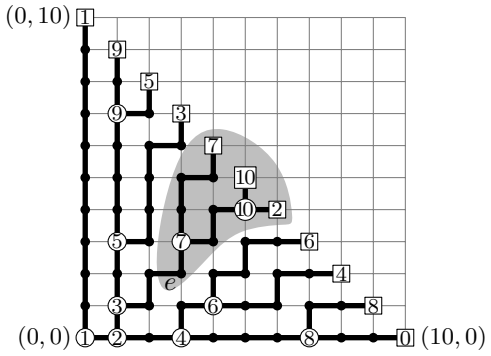


Figure 1: A spanning tree T of $\mathbf{G}(n)$ for $n = 10$. The labeled nodes are used in a low-discrepancy sequence.

to the region defined by $x + y \leq n$ in the first quadrant. From the monotonicity axiom it follows that $\mathrm{dig}(op) \subset \mathbf{G}(n)$ for any $p \in \mathbf{G}(n)$, and that dig(op) is a shortest path in the grid. We show that there exists a point $p \in \mathbf{G}(n)$ such that the Hausdorff distance $H(\mathrm{dig}(op), \overline{op})$ is $\Omega(\log n)$. Let T be the spanning tree of $\mathbf{G}(n)$ that is the union of dig(op) for all $p \in \mathbf{G}(n)$. An example spanning tree is shown in Fig. 1.

We use a classical result on pseudo-random number generation [3, 5]. Consider an infinite sequence $X = x_0, x_1, x_2, \ldots$ of real numbers in $[0, 1]$. For any given $a \in [0, 1]$ and any natural number m define $X_m(a) = |\{0 \leq i \leq m \mid x_i \in [0, a]\}|$. The *discrepancy* of the subsequence $x_0, x_1, \ldots, x_m$ is defined as $\sup_{a \in [0,1]} |am - X_m(a)|$. We use discrepancy theory in the form of the following theorem.

Theorem 1 (Schmidt [4]) *Given a sequence $X = x_0, x_1, x_2 \ldots$ of real numbers in $[0, 1]$ and a sufficiently large integer n, there exists an integer $m < n$ and a real number $a \in [0, 1]$ such that the subsequence $x_0, x_1, \ldots, x_m$ satisfies that $|am - X_m(a)| > c \log n$, where c is a positive constant independent of n.*

For $m = 1, 2, \ldots n+1$, let $L(m) = \{(i, m-1-i) \mid i = 0, \ldots, m-1\}$ be the subset of $\mathbf{G}(n)$ satisfying $x + y = m - 1$. Since there is no leaf of T in $L(m)$ for $m \leq n$ we must have exactly one branching node of degree 3 and $m - 1$ nodes of degree 2 in $L(m)$ in order to connect the m points of $L(m)$ to the $m + 1$ points in $L(m + 1)$.

We associate the leaf $(j, n - j) \in L(n + 1)$ to the number j/n and define the set $N = \{j/n : j = 0, 1, \ldots n\} \subset [0, 1]$. For each edge e of T in $\mathbf{G}(n)$, the set of vertices of $L(n + 1)$ in the subtree rooted at e forms an interval $I(e) \subset N$. Let $x(e)$ denote the largest element in $I(e)$. An example is given in Fig. 1, where $I(e) = \{0.4, 0.5, 0.6\}$ and $x(e) = 0.6$.

For a given spanning tree T we create a sequence $X(T)$ of values $x(e)$ for certain edges e. The lower bound on the discrepancy of $X(T)$ is used to show the following theorem.

Theorem 2 *For any spanning tree T there is a grid point $p \in L(n+1)$ and q in $\mathbf{G}(n)$ such that q is on the path* dig(op) *in T and the L_∞ distance from q to the line segment $\overline{op}$ exceeds $c \log n - 1$, where c is the constant considered in Theorem 1.*

Proof. For $m = 1, \ldots, n+1$ let $x_m = x(e_m)$, where e_m is the upper (i.e. vertical) branch of the unique branching node in $L(m)$. We artificially set $x_0 = 1$. Thus we obtain a sequence $X(T) = x_0, x_1, \ldots, x_n$, which is a permutation of N. Let $E(m)$ be the set of edges in T going from $L(m)$ towards $L(m+1)$. The following lemmas are obvious from the definitions:

Lemma 3 *The set $\{x(e) : e \in E(m)\}$ is equal to the set $\{x_0, x_1, \ldots, x_m\}$.*

Lemma 4 *Let e and f be edges in $E(m)$. If e is to the left of f then $x(e) < x(f)$.*

For example, the tree T in Fig. 1 creates the following sequence: $X(T) = 1, 0,$ 0.6, 0.3, 0.8, 0.2, 0.7, 0.4, 0.9, 0.1, 0.5. The labels in Fig. 1 show the correspondence between the unique internal branching node in $L(i)$ and the leaf located at $(nx_i, n - nx_i)$ in $L(n+1)$ that is associated to the number x_i. For each $i = 1, \ldots, n$ the corresponding nodes are labeled by i. In other words, each branching node and the rightmost leaf in the subtree of the upper branch of that node have the same numbering.

We now consider the discrepancy of $X(T)$. From Theorem 1, we have $0 \le a \le 1$ and $m < n$ for n large enough such that $|am - X_m(a)| > c \log n$. The following two cases should be considered:

Case 1: $X_m(a) > am + c \log n$**.** Consider the node q located at $(X_m(a)-1, m-(X_m(a)-1)) \in L(m+1)$, and let e be the edge between q and its parent in T. By definition, q is on the path dig(op) from o to the node $p = (x(e)n, n - x(e)n) \in L(n+1)$. Because of the definition of $X_m(a)$ and Lemma 3, we have exactly $X_m(a)$ edges $f \in E(m)$ for which $x(f) \le a$. However, there are also exactly $X_m(a)$ edges of $E(m)$ to the left of e, including e itself, since q is the $X_m(a)$-th node in $L(m+1)$ counted from the left. Lemma 4 implies that no edge g to the right of e can attain $x(g) \le a$. Thus, e itself must satisfy $x(e) \le a$. Now, consider the L_∞ distance of the line segment $\overline{op}$ and q. The line op goes through $(x(e)m, m - x(e)m)$, which is the L_∞-nearest point from q on op. The L_∞ distance is $(X_m(a) - 1 - x(e)m) \ge (X_m(a) - 1 - am) > c \log n - 1$.

Case 2: $X_m(a) < am - c \log n$**.** Consider the node q located at $(X_m(a), m - X_m(a)) \in L(m+1)$ and the edge e between q and its parent. Since there are only $X_m(a)$ edges $f \in E(m)$ for which $x(f) \le a$ we have $x(e) > a$ (again by Lemma 4). Node q is on the path dig(op) to $p = (x(e)n, n - x(e)n)$. Similarly to Case 1, we can show that the L_∞ distance from q to $\overline{op}$ is greater than $c \log n$. This proves the theorem. □

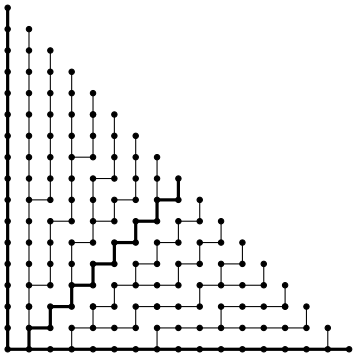


Figure 2: The spanning tree $DT(2)$ in $\mathbf{G}(n)$.

3 The upper bound results

As for the upper bound, we only give the flavor here and refer to the full version of this paper, which also gives a higher dimensional construction. We construct a spanning tree DT(2) of G, such that for every $p = (i, j) \in V$, the unique path from p to o in DT(2) defines the digital ray dig(op) simulating the line segment $\overline{op}$. This is illustrated in Fig. 2. The construction is recursive: We consider the diagonal (bold) center path. In the part below the center path, every edge in $E(2k-1)$ is horizontal for $k = 1, 2, \ldots$. As for the edges in $E(2k)$ below the center path, we copy the structure of $E(k)$. The part above the center path is constructed in a similar way.

The set of digital rays defined by DT(2) is consistent, and for any grid point $p \in \mathbf{G}(n)$, the L_∞-Hausdorff distance between dig(op) and $\overline{op}$ is less than $1 + \log n$.

The tree DT(2) is related to a famous low discrepancy sequence called van der Corput sequence [5]. Assume that n is a power of 2, and construct the sequence $X(DT(2))$ using the method of Section 2 (ignoring $x_0 = 1$). Then, we have $x_1 = 0$, $x_2 = 1/2$, $x_3 = 1/4$, $x_4 = 3/4$, and in general, if $b_1 b_2 b_3 \ldots b_s$ is the 2-adic expansion of $i-1$, $x_i = 0.b_s b_{s-1} \ldots b_1$ in 2-adic decimal expansion for $1 \le i \le n$. This sequence is indeed the van der Corput sequence.

Surprisingly, if we omit the monotonicity axiom (R5), the lower bound does not hold, and we can give a constant upper bound on the Hausdorff distance. The digital ray that we construct is locally snake-like almost everywhere; however, considered from some distance it can approximate a line segment well.

The idea is as follows: We first consider a coarse grid of width 2, and construct a spanning forest T_1 of it allowing internal leaves. Then, we replace each node v of this tree by four nodes in the original unit-width grid such that v is located in the center of gravity of these four nodes. In the last step, we convert the forest T_1 into a tree T_2 in the original unit-width grid.

Let $c > 1$ be an irrational constant. The forest

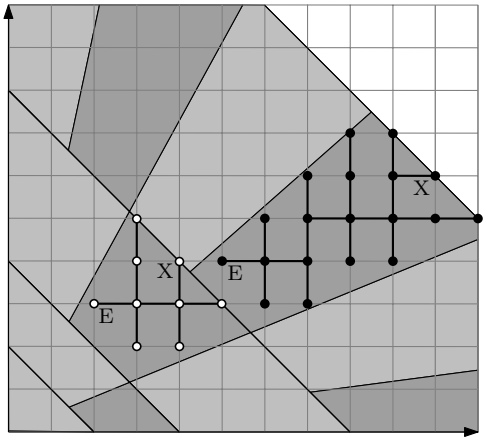


Figure 3: Trapezoid decomposition and two trees of the forest T_1.

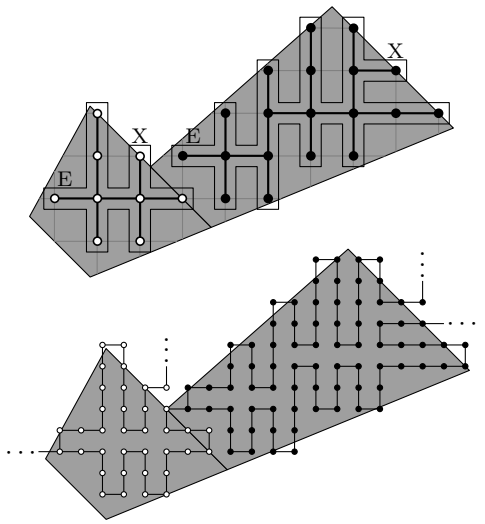


Figure 4: Walks around the two trees (top) and the corresponding part of the tree T_2 (bottom).

T_1 is constructed as follows: We consider the belt $R(k) \supset \mathbf{G}(2^{k+1}) \setminus \mathbf{G}(2^k)$ defined by $2^k < x+y \leq 2^{k+1}$ in the first quadrant, and subdivide it into trapezoids by lines $\ell_t : y = \frac{2^k - tc}{tc} x$ passing through the non-grid points $(tc, 2^k - tc)$ on the line $x + y = 2^k$ for $t = 1, 2, \ldots, \lfloor 2^k/c \rfloor$. The widths of the two parallel edges of each trapezoid are (at most) $\sqrt{2}c$ and $2\sqrt{2}c$, respectively. Further, each trapezoid F is adjacent to one trapezoid $p(F)$ in $R(k-1)$ called the parent of F, and two trapezoids $l(F)$ and $r(F)$ in the belt $R(k+1)$ that are called the left and right child, respectively. Let q be the intersection of $x + y = 2^{k+1}$ and the dividing line of $l(F)$ and $r(F)$. The nearest grid point to q in F is called the exit node of F, and the nearest grid points to q in $l(F)$ and $r(F)$ are called their entry nodes. Each trapezoid has exactly one entry and one exit node. In Fig. 3, the entry and exit nodes are marked by "E" and "X", respectively.

By gathering these trapezoids for all $k \geq \lceil \log c \rceil$, we have a decomposition of the first quadrant of the plane. Since $c > 1$, each trapezoid is wide enough so that the induced subgraph of the grid points in a trapezoid is connected. It is easy to find a spanning tree of the vertices in each trapezoid consisting of a *stem* that is shortest path from its entry node to its exit node, together with branches such that the length of each branch (i.e., the path length from the stem to the furthest leaf) is at most $2c$ as seen in Fig. 3. This gives a forest T_1 consisting of small trees, one in each trapezoid. Now, let's convert T_1 to T_2 as shown in Fig. 4. Each node of T_1 is replaced by four nodes at the corners of the surrounding unit square. Thus, we can realize the walk around each subtree of T_1 in a trapezoid F as a Hamiltonian cycle in the finer grid. We cut the cycle at the exit node, and connect it to the entry nodes of the trees in the two child trapezoids $l(F)$ and $r(F)$ as in Fig. 4. We obtain a tree T_2 that has no internal leaf.

Theorem 5 *If the monotonicity axiom (R5) is not considered, the tree T_2 defined above gives a system of digital rays in the plane grid such that the Hausdorff distance between each digital ray and its corresponding Euclidean line segment is $O(1)$.*

Proof. For any grid point p in a trapezoid F, the line segment $\overline{op}$ is contained in the union of the ancestor trapezoids of F, and also all ancestors of p in the tree T_2 are in the same union of trapezoids. Since the width of each trapezoid is at most $2\sqrt{2}c$, the distance from any point q in the path $\mathrm{dig}(op)$ in T_2 to the line op is at most $2\sqrt{2}c$. It might happen that the nearest point from q to the line op is not in the segment $\overline{op}$ since we do not assume the monotonicity axiom. However, since the length of each branch of a subtree in T_1 is at most $2c$, the Hausdorff distance between the segment $\overline{op}$ and the path from o to p in the tree is at most $(2\sqrt{2} + 2)c$. □

References

[1] D. Z. Chen, J. Chun, N. Katoh, and T. Tokuyama. Efficient algorithms for approximating a multi-dimensional voxel terrain by a unimodal terrain. In *Proc. 10th Ann. Int. Conf. Computing and Combinatorics (COCOON'04)*, volume 3106 of *Lecture Notes in Comput. Sci.*, pages 238–248, 2004.

[2] R. Klette and A. Rosenfeld. Digital straightness—a review. *Discrete Appl. Math.*, 139(1–3):197–230, 2004.

[3] J. Matoušek. *Geometric Discrepancy: An Illustrated Guide*. Springer Verlag, 1999.

[4] W. M. Schmidt. Irregularities of distribution, VII. *Acta Arithmetica*, 21:45–50, 1972.

[5] W. M. Schmidt. *Lectures on Irregularities of Distribution*. Tata Inst. Fund. Res. Bombay, 1977.

Matching a Straight Line on a Two-Dimensional Integer Domain

Emilie Charrier* Lilian Buzer†

Abstract

A two-dimensional integer domain B is defined by $\{(x,y) \in \mathbb{Z}^2 | h \leq x \leq h+D, h' \leq y \leq h'+D\}$ where h, h' and D are positive integer values. Let us consider a straight line L of the form $ax+by=c$ with real coefficients. A straight line L' of the form $a'x+b'y=c'$ is equivalent to L relative to a domain B iff the upper and the lower half planes associated with L and L' contain the same grid points in B. This paper introduces a new method to transform a straight line L into an equivalent one L' relative to B such that L' passes through at least two grid points of the domain. Our method achieves an $O(\log(D))$ time complexity. It uses elementary statements from number theory and computational geometry.

1 Introduction

We consider a two-dimensional integer domain defined by $\{(x,y) \in \mathbb{Z}^2 | h \leq x \leq h+D, h' \leq y \leq h'+D\}$ where h, h' and D are positive integer values and we want any straight line to pass through at least two grid points in the domain. Given a straight line L of the form $ax+by=c$ with real coefficients, a straight line L' is *equivalent* to L relative to this domain iff the upper and the lower half planes associated with L and L' contain the same grid points of the domain. Moreover, the straight line L' is *reduced* relative to the domain iff L' passes through at least two grid points of the domain. This reduction consists in transforming the straight line L into an equivalent one L' such that the absolute value of the coefficients of L' does not exceed D. Considering w.l.o.g. that the coefficients of the straight line L satisfy $|a| \leq |b|$, it is equivalent to reducing the straight line L relative to the domain K defined by $\{(x,y) \in \mathbb{Z}^2 | h \leq x \leq h+D\}$. Such a reduction is useful to compress the size of the coefficients of a straight line such that their absolute values do not exceed the size of the screen. For this, we compute the convex hulls of grid points of the domain K included in the upper (resp. lower) half-plane associated with L. Then, to determine a reduced straight line of L, we can compute one of the two *critical support lines*[1] of these two convex hulls. As each convex hull has at most $1+\log(D+1)$ vertices, this computation takes $O(\log(D))$ time (see [6] and [7]).

The best worst-case time complexity known to the authors to compute these integer convex hulls is $O(\log(\max\{|a|,|b|\}))$. This method uses properties of Klein polygonal lines (see [2]). Unfortunately, it only works when the coefficients of the straight line are integer. Thus, we propose a new method which computes the integer convex hulls in $O(\log(D))$ time even if the coefficients of the straight line are irrationals.

In Sec. 2, we recall some notions from number theory used in the algorithm we present. Then, we briefly introduce the existing approaches to compute the integer convex hulls in Sec. 3. Finally, we describe our method and make its complexity analysis in Sec. 4.

2 Number theory

2.1 Bezout's identity

In number theory, the *Bezout's identity* is a linear Diophantine equation. It states that if a and b are non-zero integers, then there exist two integers x and y such that $ax+by=\gcd(a,b)$ where $\gcd(a,b)$ denotes the greatest common divisor of a and b. More generally, the linear Diophantine equation $ax+by=c$ admits integer solutions iff the gcd of a and b divides c. The set of solutions is given by:

$$\left\{ \left(x_0 + \frac{kb}{\gcd(a,b)},\ y_0 - \frac{ka}{\gcd(a,b)} \right) \mid k \in \mathbb{Z} \right\}$$

where (x_0, y_0) denotes a particular solution.

You can consult [4] for more details on Bezout's identity.

A vector $u=(u_1,u_2)$ with integer coordinates is called *irreducible* if $\gcd(u_1,u_2)$ is equal to one. Let $u=(u_1,u_2)$ and $v=(v_1,v_2)$ denote two vectors, the value $u \wedge v$ is equal $u_1v_2-u_2v_1$. It corresponds to the z component of the cross product of u and v and it is equal to the signed area of a parallelogram generated by u and v.

Definition 1 *Let u denote an irreducible vector. A Bezout vector of u is a vector v with integer coordinates such that $u \wedge v = 1$. This means that the parallelogram $(0, u, u+v, v)$ contains no grid points in its interior.*

*Laboratory CNRS-UMLV-ESIEE, UMR 8049, DGA/D4S/MRIS charriee@esiee.fr

†Laboratory CNRS-UMLV-ESIEE, UMR 8049, buzerl@esiee.fr

[1] Critical support lines of two convex polygons are straight lines tangent to both polygons, such that the polygons lie on opposite sides of the lines.

2.2 Continued fractions

2.2.1 Definitions and properties

We present the definition of *continued fractions* and some of their properties (see [1] for a more detailed introduction).

Definition 2 *The simple continued fraction decomposition of a real number x corresponds to:*

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{\cdots}}}}$$

where a_0 denotes an integer value and where a_i denotes a strictly positive integer value for all $i \geq 1$. We usually use the notation $x = [a_0, a_1, a_2, \cdots]$.

The *principal convergents* of x correspond to its rational approximations p_k/q_k. The two convergents p_0/q_0 and p_1/q_1 are respectively set to 0/1 and 1/0 whereas the others are obtained by truncating the continued fraction decomposition after the k-th partial quotient. The numerator and the denominator of the principal convergents are computed as follows:

$$\begin{cases} p_0 = 0 & p_1 = 1 & p_{k+2} = p_k + a_k p_{k+1} \\ q_0 = 1 & q_1 = 0 & q_{k+2} = q_k + a_k q_{k+1} \end{cases}$$

Each convergent of even order is less than the whole continued fraction and each convergent of odd order is greater than the whole continued fraction. Each convergent is closer in value to the whole continued fraction than the preceding. The *intermediate convergents* between two principal convergents p_k/q_k and p_{k+2}/q_{k+2} are defined as:

$$\frac{p_k + i p_{k+1}}{q_k + i q_{k+1}}, \quad i = 1 \cdots a_k - 1$$

Let S_E (resp. S_O) denote the series defined by all the principal convergents of even (resp. odd) order intercalated with their corresponding intermediate convergents. If x is a rational number then the last term of one of the two series is not equal to x. The rational number x is added to the end of this series. Let us enunciate a useful proposition (see [1] for the proof).

Proposition 1 *Let s denote the rational number which is less (resp. greater) than a real number r, which most closely approximates r and such that its denominator does not exceed an integer value d. The rational number s is the term of the series S_E (resp. S_O) of r with the greatest denominator which does not exceed d.*

Example 1 *We want to find the rational number whose denominator does not exceed 60 and which is the best approximation of 779/207. We have* $779/207 = [3, 1, 3, 4, 2, 5]$. *It follows the two series:*

$$S_E = \frac{0}{1}, \frac{1}{1}, \frac{2}{1}, \frac{3}{1}, \frac{7}{2}, \frac{11}{3}, \frac{15}{4}, \frac{79}{21}, \frac{143}{38}, \frac{779}{207}$$

$$S_O = \frac{1}{0}, \frac{4}{1}, \frac{19}{5}, \frac{34}{9}, \frac{49}{13}, \frac{64}{17}, \frac{207}{55}, \frac{350}{93}, \frac{493}{131}, \frac{636}{169}, \frac{779}{207}$$

We deduce that the rational number 143/38 *in S_E (resp.* 207/55 *in S_O) is the best approximation of* 779/207, *which is less (resp. greater) than* 779/207 *and whose denominator does not exceed 60.*

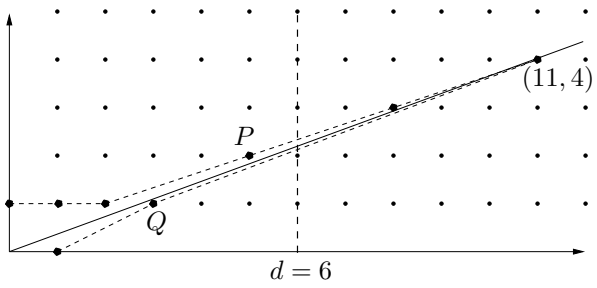


Figure 1: Example of Klein's polygonal lines for 4/11 and $d = 6$

2.2.2 Geometrical interpretation

We can establish a correspondence between a rational number a/b and an integer vector of coordinates (b, a) in the Euclidean plane. As a result, the two series S_E and S_O approximating the rational number p/q correspond to two series of integer vectors in the Euclidean plane approximating the integer vector (q, p). We can interpret Prop. 1 in the Euclidean plane. The rational number which most closely approximates the rational number p/q such that its denominator does not exceed d corresponds in the Euclidean plane to the integer vector which most closely approximates the vector (q, p) such that its abscissa does not exceed d. The two series S_E and S_O interpreted in the Euclidean plane are called *Klein's polygonal lines* (see [3]). Figure 1 shows an example of the series S_E and S_O where $p/q = 4/11$ and $d = 6$ interpreted in the Euclidean plane. In this example, $P = (5, 2)$ and $Q = (3, 1)$ correspond to the best approximations of the vector $(11, 4)$ whose abscissa does not exceed 6.

3 Existing methods to compute integer convex hulls

As introduced in Sec. 1, we have to compute the border of the convex hull of grid points located above (resp. below) L in the domain K. We call them respectively the upper and the lower convex hulls (see Fig. 2 for an example).

To compute these convex hull borders, a brute force approach would consist in considering the $D + 1$ grid

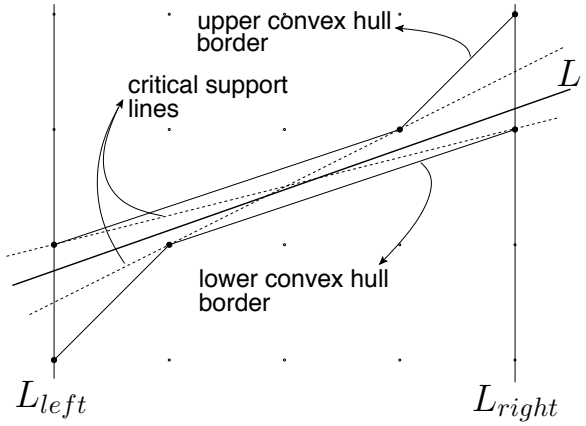


Figure 2: Lower and upper convex hulls

points located just above the straight line L and the $D+1$ grid points located just below the straight line L (digital straight lines of L). Then, we would compute their convex hulls in $O(D\log(D))$ time (see [5]). When a and b are integer values, we can also use a more efficient approach which consists in using Klein polygonal lines and Prop. 1 (see [3] and [2]). This second approach runs in $O(\log(\max\{|a|,|b|\}))$ time. In this paper, we propose a new algorithm to compute these convex hulls which runs in $O(\log(D))$ time even if a and b are irrational values.

4 Introducing a new approach

We present our approach to compute the border of the upper and of the lower convex hull of L.

4.1 Algorithm design

Let $\lfloor\cdot\rfloor$ denote the floor function. Let L_{left} and L_{right} respectively denote the straight lines of the form $x=h$ and $x=h+D$ that delineate the domain K.

4.1.1 General presentation

The computation of the upper and of the lower convex hull is done in two steps. We first successively determine the vertices $(A_i)_{1\leq i\leq n}$ (resp. $(B_j)_{1\leq j\leq m}$) of the upper (resp. lower) convex hull from left to right, starting from two grid points lying on the straight line L_{left} (see Fig. 5). When we find a vertex which does not lie in K, we stop. Then, we begin the second step by computing new vertices of the upper and of the lower convex hull from right to left, starting from two grid points lying on the straight line L_{right}. Let us describe the first part of the computation, from left to right (the second part of the computation is similar).

Suppose we know the vertices A_i and B_j of the upper and of the lower convex hulls respectively. Our approach determines in constant time the vertex A_{i+1} or the vertex B_{j+1}. For this, we look for a grid point

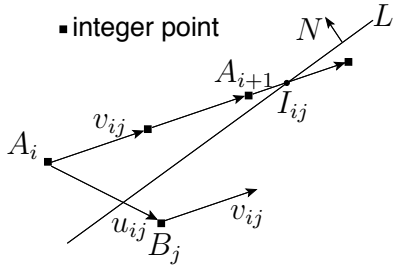


Figure 3: Example when $v_{ij}\cdot N<0$

located above or below L and such that the angle between the vectors $A_{i-1}A_i$ and A_iA_{i+1} or the angle between the vectors $B_{i-1}B_i$ and B_iB_{i+1} is minimal. In the next section, we explain how to determine this next vertex.

4.1.2 Carrying out the method

We determine the next vertex A_{i+1} or B_{j+1} of the convex hulls by linear combination of two vectors u_{ij} and v_{ij}. At each iteration, the vector u_{ij} corresponds to the vector A_iB_j and the vector v_{ij} corresponds to a particular Bezout vector of u_{ij}, defined as follows: the grid point A_i+v_{ij} is located above L and the grid point $A_i+v_{ij}+u_{ij}$ is located below L. Such a vector v_{ij} is called a *valid Bezout vector* of u_{ij} relative to L. At the first iteration, the vector u_{ij} is equal to $(0,-1)$.

Let N denote the normal vector of the straight line L such that the dot product $A_i\cdot N$ is greater than c for all vertices A_i, $1\leq i\leq n$. Let I_{ij} denote the intersection of the straight line L and the straight line of direction vector v_{ij} passing through A_i. Let J_{ij} denote the intersection of the straight line L and the straight line of direction vector v_{ij} passing through B_j. At each iteration, we look at the sign of the dot product $v_{ij}\cdot N$. When $v_{ij}\cdot N$ is strictly negative, it means that I_{ij} is equal to $A_i+\alpha v_{ij}$ where α is a positive value greater than one. By definition of Bezout vector, the triangle $A_iI_{ij}B_j$ contains no grid point in its interior. As a consequence, the grid point $A_i+\lfloor\alpha\rfloor v_{ij}$ corresponds to the vertex A_{i+1} of the upper convex hull (see Fig. 3). In the same way, when $v_{ij}\cdot N$ is strictly positive, it means that J_{ij} is equal to $B_j+\beta v_{ij}$ where β is a positive value greater than one. The triangle $B_jJ_{ij}A_i$ contains no grid point in its interior. As a result, the grid point $B_j+\lfloor\beta\rfloor v_{ij}$ corresponds to the vertex B_{j+1} of the lower convex hull (see Fig. 4).

Remarks 1 *Some special cases can occur but they are not detailed in this paper. Indeed, the case where the dot product $v_{ij}\cdot N$ is equal to zero is trivial and is left to the reader. In the same way, the case where intermediate grid points lie on the last found edge is treated very simply.*

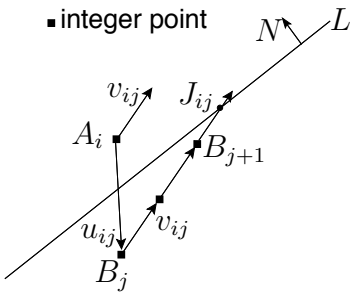


Figure 4: Example when $v_{ij} \cdot N > 0$

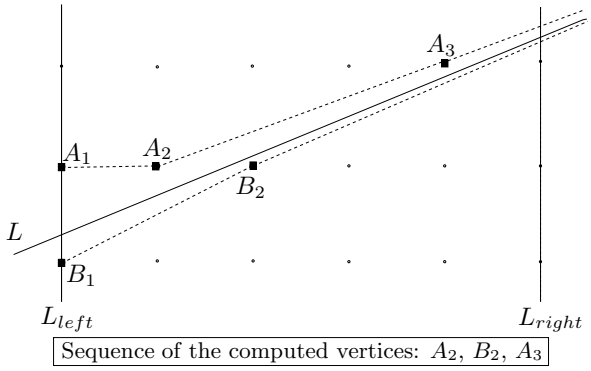


Figure 5: The first step of the computation

4.2 Complexity analysis

The first two vertices A_1 and B_1 correspond to the grid points lying on L_{left} which are closest to L respectively above and below L. We compute them in $O(1)$ time. Then, at each iteration we determine a new vertex of the upper or of the lower convex hull. For this, we use an oracle which computes in constant time the intersection of two straight lines. Moreover, we notice that the vector v_{ij} also corresponds to a Bezout Vector of the vector $u_{i+1\ j}$ or $u_{i\ j+1}$. So, we can transform the vector v_{ij} into a valid Bezout vector for $u_{i+1\ j}$ or $u_{i\ j+1}$ using the same oracle in $O(1)$ time. As a result, each iteration runs in constant time. As each convex hull border has at most $1 + \log(D + 1)$ vertices (see [7]), our algorithm computes each convex hull in $O(\log(D))$ time. Algorithm 4.2 is a simplified overview of our algorithm. Indeed, we show only the main steps and we consider that $v_{ij} \cdot N$ is not equal to zero. Moreover, we also consider that the straight line L does not pass through grid points.

5 Conclusion

We describe a method to reduce the straight line L of the form $ax + by = c$ into an equivalent one relative to a domain K of size D such that the reduced straight line passes through at least two grid points of the domain. For this, we compute the border of the convex hull of the grid points lying in the domain

Algorithm 1 Simplified overview of the algorithm

```
1 Init(A_1,B_1,u_11,v_11)
2 i ← 1 j ← 1
3    WHILE (A_i and B_j on the left of L_right)
4       IF(v_ij · N < 0)
5          ⌊β⌋ ← Intersection(A_i, v_ij, L)
6          A_{i+1} ← A_i + ⌊β⌋v_ij   i ← i + 1
7       ELSE
8          ⌊β⌋ ← Intersection(B_j, v_ij, L)
9          B_{j+1} ← B_j + ⌊β⌋v_ij   j ← j + 1
10      update(u_ij,v_ij)
```

which are located above and below the straight line. Each critical support line of the two convex hulls corresponds to a reduced straight line of L. We prove that our method runs in $O(\log(D))$ time contrary to the other approaches which run in $O(D \log(D))$ time or in $O(\log(\max\{|a|, |b|\}))$ time. Our algorithm runs even if the coefficients of the straight line are irrational numbers. Moreover, when we compute the lower and the upper convex hulls, we know the grid point of the domain K which is closest to the straight line L. Interpreted as a rational number, this grid point corresponds to the rational number whose denominator is greater than h and less than $h + D$, which best approximates the slope of L.

References

[1] G. Chrystal. *Algebra- An elementary text-book Part II.* Adam and Charles Black, Edinburgh, 1889, Chapter XXXII, 423-452.

[2] W. Harvey. *Computing two-dimensional integer hulls.* SIAM J. Compute., **28(6)**, 1999, 2285-2299.

[3] F. Klein. *Ausgewählte Kapitel der Zahlentheorie.* Teubner, 1907.

[4] T. Mora. *Solving Polynomial Equation Systems I: The Kronecker-Duval Philosophy.* Encyclopedia of Math. and its applications, **88** Cambridge University Press, 2002.

[5] S.S. Skiena. *Convex hull.* The algorithm design manual, N. Y. Springer-Verlag, §8.6.2, 1997, 351-354.

[6] G.T. Toussaint. *Solving geometric problem with the rotating calipers.* Proceedings of IEEE MELECON'83, Greece, pp. A10.02/1-4, 1983.

[7] N.Y. Zolotykh. *On the number of vertices in integer linear programming problems.* Technical report, University of Nizhni Novgorod 2000.

Exploring Simple Triangular and Hexagonal Grid Polygons Online

Daniel Herrmann* Tom Kamphans** Elmar Langetepe*

Abstract

We examine online grid covering with hexagonal and triangular grids. For arbitrary environments without obstacles we provide a strategy that produces tours of length $S \leq C + \frac{1}{4}E - \frac{5}{2}$ for hexagonal grids, and $S \leq C + E - 4$ for triangular grids. C is the number of cells—the area—, E is the number of boundary edges—the perimeter—of the environment. We show that our strategy is $\frac{4}{3}$-competitive, and give lower bounds of $\frac{14}{13}$ ($\frac{7}{6}$) for hexagonal (triangular) grids.

Keywords: Exploration, covering, grid graphs

1 Introduction

Exploring an unknown environment is one of the basic tasks of autonomous mobile robots. For some applications such as robots with limited vision or robots that have to visit every part of the environment (lawn mowers, cleaners), it is convenient to subdivide the given environment by a regular grid into *cells*. The robot's position is always given by the cell currently occupied by the robot. From its current position, it knows the neighboring cells and it can move to a free one. The robot's task is to visit every free cell and to return to the start.

There are three regular tilings: square, hexagonal, or triangular subdivisions. We call the subdivisions of the given environment a square polygon (hexagonal polygon, triangular polygon; respectively). Hexagonal cells are a matter of particular interest for robots that are equipped with a circular tool such as lawn mowers, because hexagonal grids provide a better approximation for the tool than square grids [1].

In a square polygon with obstacles, the offline problem (i.e., finding a minimum length tour that visits every cell) is NP-hard [13], but there are $1 + \epsilon$ approximation schemes (e.g., [3]). For square polygons there is a $\frac{53}{40}$ approximation by Arkin et al. [1].

In a square polygon without obstacles, the complexity of constructing offline a minimum length tour is still open. There are approximations with factors $\frac{4}{3}$ [14] and $\frac{6}{5}$ [1]. There is an $O(C^4)$ algorithm for deciding the existence of Hamiltonian cycles in a square grid. Hamiltonian paths were considered by Everett [6]. HCs on triangular and hexagonal grids were studied by Arkin et al. [2], and Islam et al. [12].

Our interest is in the *online* version of the cell exploration problem for *hexagonal* and *triangular* polygons. Square polygons with holes were considered by Gabriely and Rimon [7] and Icking et al. [11]. Our exploration strategy is based on the $\frac{4}{3}$-competitive strategy *SmartDFS* by Icking et al. [10], which needs at most $\#\text{Cells} + \frac{\#\text{Edges}}{2} - 3$ steps from cell to cell. Also, there is a lower bound of $\frac{7}{6}$ on the competitive factor.

Subdividing the robot's environment into grid cells is used also in the robotics community (e.g., [4]). See also the survey by Choset [5].

2 Preliminaries

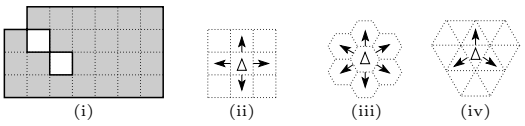


Figure 1: (i) A simple grid polygon, (ii)-(iv) neighboring (arrows) and touching cells.

We consider polygons that are subdivided by a regular grid into *cells*. A cell is *free* if it can be visited by the agent, otherwise *blocked*. We call two cells *neighboring* if they share a common edge, *touching* if they share a common corner. A *path* is a sequence of consecutively neighboring cells, a *grid polygon* is a path-connected set of free cells. A polygon without blocked cells inside its boundary is called *simple*. From its current position, the agent can find out which neighbor is free and which one is blocked, and it can move in one *step* to a free neighbor, see Fig. 1. The agent has enough memory to store a map of known cells.

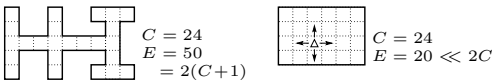


Figure 2: E distinguishes *thin* and *thick* polygons.

We analyze the performance of an online exploration strategy using the *area*, C (the number of cells), and the *perimeter*, E (the number of boundary edges). E is adequate to distinguish between thin environments that have many corridors of width 1, and thick environments that have wider areas; see Fig. 2.

*University of Bonn, Institute of Computer Science I, 53117 Bonn, Germany.
**Braunschweig University of Technology, Computer Science, Algorithms Group, 38106 Braunschweig, Germany

3 Lower bounds

First, our interest is in the best competitive factor we can expect for an online strategy that visits every cell and returns to the start cell.

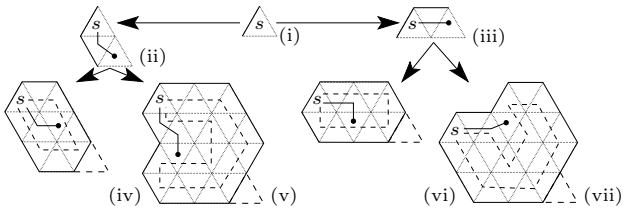


Figure 3: A lower bound on the exploration of simple triangular polygons (thin dashed: optimal solution, bold dashed: next block).

Theorem 1 *There is no online strategy for the exploration of simple triangular (hexagonal) grid polygons with a competitive factor better than* $\frac{7}{6}$ *(*$\frac{14}{13}$*, resp.).*

Proof. For triangles, we start in a cell with two neighbors, see Fig 3(i). If the first step is to the south, we add a cell such that the only possible step is to the southwest (3(ii)). Otherwise, we force a step to the east (3(iii)). In both cases, the agent has the choice to leave or to follow the polygon's boundary. In either case we close the polygon. In the first case, the agent needs at least 12 steps while the optimal path has 10 steps (3(iv),(vi)). In the second case, the agent needs $\geq$ 26 steps; the optimal path needs 22 steps (3(v), (vii)). To construct arbitrarily large polygons, we use more of these blocks and glue them together, the cell shown with bold dashed lines is the next start cell. Unfortunately, both the online strategy and the optimal path need two additional steps for the transition. For n blocks we have in the best case a ratio of $\frac{26+28(n-1)}{22+24(n-1)}$, which goes to $\frac{7}{6}$ if n goes to infinity.

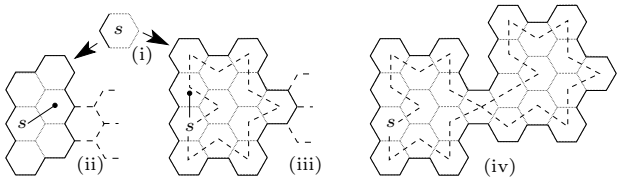


Figure 4: A lower bound for the exploration of simple polygons. The dashed lines show the optimal solution.

For hexagons, we start in a cell with four neighbors, see Fig. 4(i). The agent may leave the polygon's boundary with a step NW or SW, or follow the boundary by walking north or south. In either case we fix the polygon (Fig. 4(ii),(iii)), yielding a ratio of $\frac{7}{6}$ or $\frac{13}{12}$, respectively. The subsequent block attaches using the cell(s) shown with bold dashed lines. Again, we need one or two additional steps for the transition (4(iv)), yielding a best-case ratio of $\frac{13+14(n-1)}{12+13(n-1)} \longrightarrow \frac{14}{13}$. □

4 Exploring simple polygons

In this section, we briefly describe the exploration strategy *SmartDFS* [11]. As a first approach, we can apply a depth-first search (DFS): We explore the polygon following the left-hand rule; that is, for every entered cell the agent tries to continue its path to a neighboring unexplored cell in clockwise order. This results in a complete exploration, but takes $2C - 2$ steps. We introduce two improvements.

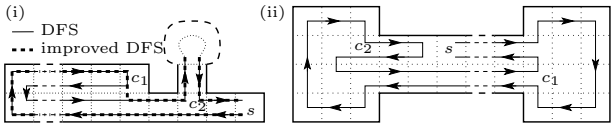


Figure 5: Improvements to DFS: (i) optimize return path, (ii) detect polygon splits.

First, we return directly to those cells that have unexplored neighbors. See Fig. 5(i): DFS walks from c_1 to c_2 through the completely explored corridor. A more efficient strategy walks on a shortest path (on known cells) from c_1 to c_2.

In Fig. 5(ii), DFS walks four times through the corridor. A more clever solution explores the right part immediately after the first visit of c_1, and continues with the left part, resulting in only two visits. When c_1 is explored, the graph of unvisited cells splits into two components. We call cells with this property *split cells*. The second improvement is to handle split cells.

SmartDFS(*P*, *start*):
  Choose *dir*, such that reverse(*dir*) is blocked;
  ExploreCell(*dir*);
  Walk on the shortest path to *start*;

ExploreCell(*dir*):
  base := current position;
  if not isSplitCell(*base*) **then**
    forall neighbors *c* of *base*, in clockwise order
      ExploreStep(*base*, direction towards *c*);
  else Choose different order, see Sect. 5.

ExploreStep(*base*, *dir*):
  if unexplored(*base*, *dir*) **then**
    Walk on shortest path to *base*;
    move(*dir*);
    ExploreCell(*dir*);
  end if

5 The analysis of SmartDFS

In this section, we briefly analyze the performance of our strategy in hexagonal and triangular grids. See our full version [9, 8] for the complete proofs.

To show our main theorem, we need an upper bound on the length of a path inside a grid polygon.

Lemma 2 *The length of a shortest path,* Π*, between two cells in a grid polygon is bounded by*
$|\Pi| \leq \frac{1}{4}E(P) - \frac{3}{2}$ *for hexagonal polygons, and*
$|\Pi| \leq E(P) - 3$ *for triangular polygons.*

First, we give an upper bound on the number of steps needed by our strategy. The basic idea is an induction on the number of split cells, so let P^* be the part of P that is currently explored. When we meet a split cell, c, P^* divides into three parts: $P^* = K_1 \dot\cup K_2 \dot\cup \{\text{ visited cells of } P^*\}$, where K_1 and K_2 denote the connected components of the set of unvisited cells of P^*, see Figs. 6 and 7.

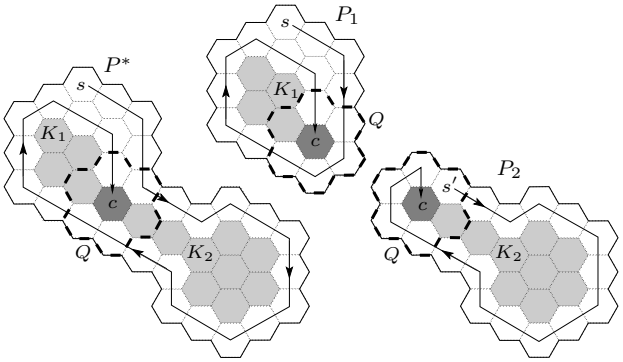


Figure 6: A split: K_1 is of type C, K_2 type A.

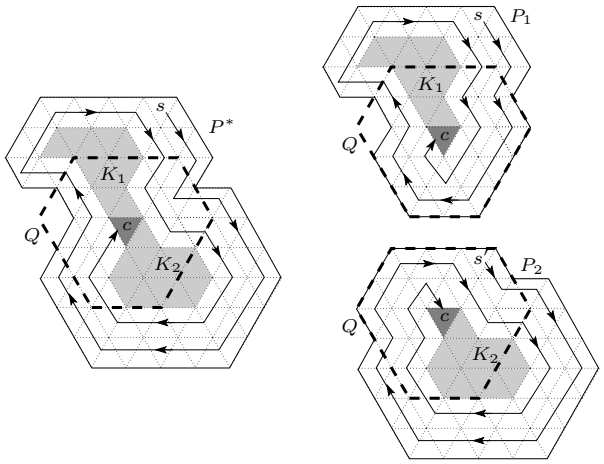


Figure 7: A split: K_1 is of type C, K_2 type B.

Now, we divide P^* into two parts, P_1 and P_2, such that each of them is an extension of the two components. Both polygons overlap in the area around the split cell c. If there is a polygon that does *not* contain s, we explore the corresponding component first, expecting that in this part the path from the last visited cell back to s is the shorter than in the other part. In the following, let K_2 denote the component that is explored first.

We can decide which component we have to visit first using the following definition:

Definition 1 *The boundary cells of* P *uniquely define the* first layer. P *without its first layer is the* 1-offset. *The* ℓ*th layer and the* ℓ*-offset are defined successively. When a split cell occurs in layer* ℓ*, every component is one of the following types:*
A. the part of layer ℓ *that surrounds* K_i *is* not *visited*
B. . . . completely *visited*
C. . . . partially *visited*

If a component of type C exists, it is reasonable to explore it at last. Otherwise, we proceed using the left-hand rule, but omit the first possible step [9].

Now, let Q be the polygon made of c extended by q layers, where $q := \ell$ if K_2 is of type B, and $q := \ell - 1$ if K_2 is of type A. We choose $P_2 \subset P^* \cup Q$ such that $K_2 \cup \{c\}$ is the q-offset of P_2, and $P_1 := ((P^* \backslash P_2) \cup Q) \cap P^*$. Further, we require $P^* \cup Q = P_1 \cup P_2$ and $P_1 \cap P_2 \subseteq Q$. The choice of P_1, P_2, and Q ensures that the paths in $P_1 \backslash Q$ and $P_2 \backslash Q$ do not change compared to P^*. The parts of the path that lead from P_1 to P_2 and back are fully contained in Q. Just the parts inside Q are bended to connect the appropriate paths inside P_1 and P_2; see Figs. 6, 7.

Theorem 3 *Let* P *be a simple grid polygon with* $C(P)$ *cells and* $E(P)$ *edges.* P *can be explored with*
$S(P) \leq C(P) + \frac{1}{4}E(P) - \frac{5}{2}$ *(hexagonal polygons)*
$S(P) \leq C(P) + E(P) - 4$ *(triangular polygons)*
steps. This bound is tight.

Proof. (sketch for hexagonal polygons)
We show by an induction on the number of split cells that the number of *additional* cell visits, $ex(P^*)$, is smaller than $\frac{1}{4}E(P^*) - \frac{5}{2}$.

In the induction base we have no split cell: SmartDFS visits every cell and returns to the start cell. Without a split, all cells of P can be visited by a path of length $C(P) - 1$. By Lemma 2, the shortest path back to s is smaller than $\frac{1}{4}E(P^*) - \frac{3}{2}$.

Now, let c be the first split cell in P^*. Two new components, K_1 and K_2, occur. Let P_1, P_2, and Q defined as earlier. There is no split in $P_2 \backslash (K_2 \cup \{c\})$, but c is visited twice. Thus, we have

$$ex(P^*) \leq ex(P_1) + ex(K_2 \cup \{c\}) + 1.$$

Now, we apply the induction hypothesis to P_1 and $K_2 \cup \{c\}$ and get

$$ex(P^*) \leq \tfrac{1}{4}E(P_1) - \tfrac{5}{2} + \tfrac{1}{4}E(K_2 \cup \{c\}) - \tfrac{5}{2} + 1.$$

The ℓ-offset, P', of P fulfills $E(P') \leq E(P) - 12\ell$ [9]:

$$ex(P^*) \leq \tfrac{1}{4}E(P_1) + \tfrac{1}{4}E(P_2) - 3q - 4.$$

With $E(P_1) + E(P_2) = E(P^*) + E(Q)$ and $E(Q) = 12q + 6$ we have

$$ex(P^*) \leq \tfrac{1}{4}E(P^*) + \tfrac{3}{2} - 4 = \tfrac{1}{4}E(P^*) - \tfrac{5}{2}\,.$$

It is easy to see that this bound is exactly achieved in corridors of width 1. The exploration of such a corridor needs $2(C(P) - 1)$ steps. On the other hand, the number of edges is $E(P) = 4C(P) + 2$. □

Theorem 4 *The strategy SmartDFS is $\frac{4}{3}$-competitive in hexagonal and triangular polygons. This factor is tight.*

Proof. (sketch for hexagonal polygons)
Corridors of width 1 or 2 play a crucial role in the following, so we refer to them as *narrow passages*. More precisely, a cell, c, belongs to a narrow passage, if c can be removed without changing the layer number of any other cell. It is easy to see that narrow passages are explored optimally: In corridors of width 1 both SmartDFS and the optimal strategy visit every cell twice, and in the other case both strategies visit every cell exactly once. Thus, we can assume that P is a polygon without narrow passages. If P has no split cell in the first layer, we can show that $E(P) \leq \frac{4}{3}\,C(P) + \frac{26}{3}$ holds ($E(P) \leq \frac{1}{3}\,C(P) + \frac{14}{3}$ for triangular polygons). Further, in these type of polygons SmartDFS needs two steps fewer than shown in Theorem 3.

Now, we show by induction on the number of split cells in the first layer that $S(P) \leq \frac{4}{3}C(P) - \frac{7}{3}$ holds.

For the induction base we can apply the observations from above: $S(P) \leq C(P) + \frac{1}{4}\,E(P) - \frac{9}{2} \leq C(P) + \frac{1}{4}\left(\frac{4}{3}\,C(P) + \frac{26}{3}\right) - \frac{9}{2} = \frac{4}{3}\,C(P) - \frac{7}{3}$.

Two cases occur if we meet a split cell, c, in the first layer: Either the new component was never visited before (type A), or we meet a visited cell, c', that touches the current cell (type B). In the first case let $Q := \{c\}$, in the second case $Q := \{c, c'\}$.

Similar to the proof of Theorem 3, we split the polygon P into two parts, both including Q. Let P'' denote the part that includes the component of type A or B, P' the other part. For $|Q| = 1$ we conclude $S(P) = S(P') + S(P'')$ and $C(P) = C(P') + C(P'') - 1$. Applying the induction hypothesis to P' and P'' yields $S(P) = S(P') + S(P'') < \frac{4}{3}\,C(P) - \frac{7}{3}$.

For $|Q| = 2$ we gain some steps by merging the polygons. If we consider P' and P'' separately, we count the steps from c' to c—or vice versa—in both polygons, but in P the path from c' to c is replaced by the exploration path in P''. Thus, we have $S(P) = S(P') + S(P'') - 2$ and $C(P) = C(P') + C(P'') - 2$. Applying the induction hypothesis yields our claim.

OPT needs at least C steps, which, altogether, yields a competitive factor of $\frac{4}{3}$. This factor is achieved in a corridor of width 3, see Fig. 8. □

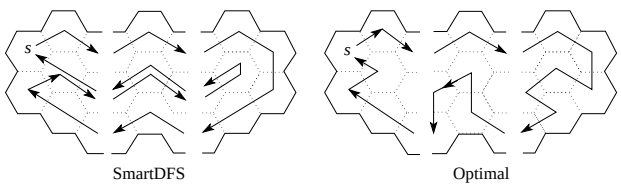


Figure 8: $S(P) = \frac{4}{3}S_{\mathrm{Opt}}(P) - \frac{7}{3}$ holds.

6 Summary

We considered the online exploration of hexagonal and triangular grid polygons with SmartDFS.

For hexagonal polygons we gave a lower bound of $\frac{14}{13}$ and showed that SmartDFS explores polygons with C cells and E edges using no more than $C + \frac{1}{4}E - \frac{5}{2}$ steps. For triangular polygons we have a lower bound of $\frac{7}{6}$ (matching the lower bound for square polygons) and an upper bound of $C + E - 4$ on the number of steps. Further, we showed that both strategies are $\frac{4}{3}$-competitive. An interesting open problem is how to close the gap between the upper bound and the lower bound on the competitive factors.

References

[1] E. M. Arkin, S. P. Fekete, and J. S. B. Mitchell. Approximation algorithms for lawn mowing and milling. *Comput. Geom. Theory Appl.*, 17:25–50, 2000.

[2] E. M. Arkin, J. S. B. Mitchell, and V. Polishchuk. Two new classes of hamiltonian graphs. In *Proc. Europ. Conf. Comb. Graph Theo. Appl.*, volume 29C of *e-Notes Discr. Math*, pages 565–569, 2007.

[3] S. Arora. Polynomial time approximation schemes for Euclidean TSP and other geometric problems. In *Proc. 37th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 2–11, 1996.

[4] A. M. Bruckstein, M. Lindenbaum, and I. A. Wagner. Mac vs. pc - determinism and randomness as complementary approaches to robotic exploration of continuous unknown domains. *Internat. J. Robotics Res.*, 19(1):12–31, 2000.

[5] H. Choset. Coverage for robotics - A survey of recent results. *Ann. Math. Artif. Intell.*, 31:113–126, 2001.

[6] H. Everett. Hamiltonian paths in non-rectangular grid graphs. Report 86-1, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, 1986.

[7] Y. Gabriely and E. Rimon. Competitive on-line coverage of grid environments by a mobile robot. *Comput. Geom. Theory Appl.*, 24:197–224, 2003.

[8] D. Herrmann. Exploration von Dreiecks– und Sechseckgittern. Diplomarbeit, University of Bonn, January 2008.

[9] D. Herrmann, T. Kamphans, and E. Langetepe. Exploring simple triangular and hexagonal grid polygons online. Technical Report 007, Department of Computer Science I, University of Bonn, 2007. http://web.informatik.uni-bonn.de/I/publications/hkl-esthg-07.pdf.

[10] C. Icking, T. Kamphans, R. Klein, and E. Langetepe. Exploring grid polygons online. Technical Report 001, Department of Computer Science I, University of Bonn, December 2005. http://web.informatik.uni-bonn.de/I/publications/ikkl-egpol-05.pdf.

[11] C. Icking, T. Kamphans, R. Klein, and E. Langetepe. Exploring simple grid polygons. In *11th Internat. Comput. Combin. Conf.*, volume 3595 of *Lecture Notes Comput. Sci.*, pages 524–533. Springer, 2005.

[12] K. Islam, H. Meijer, Y. N. Rodríguez, D. Rappaport, and H. Xiao. Hamilton circuits in hexagonal grid graphs. In *Proc. 19th Canad. Conf. Comput. Geom.*, pages 85–88, 2007.

[13] A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter. Hamilton paths in grid graphs. *SIAM J. Comput.*, 11:676–686, 1982.

[14] S. Ntafos. Watchman routes under limited visibility. *Comput. Geom. Theory Appl.*, 1(3):149–170, 1992.

Manifold Homotopy via the Flow Complex

Bardia Sadri*

Abstract

It is known that the critical points of the distance function induced by a dense sample P of a submanifold Σ of $\mathbb{R}^n$ are distributed into two groups, one lying close to Σ itself, called the *shallow*, and other close to medial axis of Σ, called *deep* critical points. We prove that under (uniform) sampling assumption, the union of stable manifolds of the shallow critical points have the same homotopy type as Σ itself and the union of the stable manifolds of the deep critical points have the homotopy type of the complement of Σ. The separation of critical points under uniform sampling entails a separation in terms of distance of critical points to the sample. This means that if a given sample is dense enough with respect to two or more submanifold of $\mathbb{R}^n$, the homotopy type of all such submanifolds as well as that of their complements are captured as unions of stable manifolds of shallow critical points, in a filtration of the flow complex based on the distance of critical points to sample.

1 Introduction

The *flow complex* was introduced by Giesen and John [9] as a tool for geometry modeling. Much of the mathematical foundations behind the flow complex were well-explored; see [10] and references therein. Further important properties of the *flow map* induced by a generalized gradient of the distance function induced by compact sets have been subject of recent investigations, see e.g. [12]. In [8], it was noted empirically that the flow complex derived from a dense sample of a surface, though often much coarser than the Delaunay complex of the same point set, does contains a subcomplex that *approximates* the surface much in the same was as the Delaunay complex.

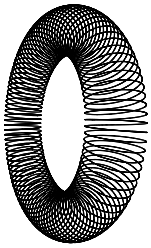
Surface reconstruction is the problem of producing from a discrete sample of a surface Σ a concisely represented surface $\tilde{\Sigma}$ that closely approximates Σ and shares its topology, provided that the sample is dense enough. This problem has a rich literature spanning several disciplines; see [2] for a survey of Delaunay-based algorithms which have particularly been the most successful in providing geometric and topological guarantees. Traditionally, "topological equivalence" is interpreted as *homeomorphism* or even *ambient isotopy*. This in particular requires the reconstructed object to also be a manifold and of the same dimension as the target surface. In this paper, we relax this interpretation to *homotopy equivalence* (See [11] for definitions). In other words, we seek to capture the homotopy type of a manifold by finding a topological space that that is not necessarily a manifold but approximates the original manifold in Hausdorff distance.

Prior to [8], flow methods were employed in surface reconstruction (e.g. [7]) but the first of such algorithms with geometric and topological guarantees was found by Dey et al. [5] who proved a sharp separation of critical points of the distance function induced by surface samples into two groups one lying close to the surface (shallow) and the other close to its medial axis (deep). They further showed that in 3D, the boundary of the union of stable manifolds of inner or outer deep critical points is homeomorphic to the original surface, provided that the sample is dense enough and *tight*. However, this does not generalize to higher dimensions. This paper aims to achieve this, albeit with certain modifications. On the down-side, we use *uniform* sampling (as opposed to adaptive sampling used in [5]) although we relax the tightness requirement. Moreover, homeomorphism is weakened to homotopy equivalence. On the upside, we prove that the union of stable manifolds of shallow critical points approximates the manifold and captures its topology while that of deep ones does the same for the complement of the manifold. Plus, we show that this works for any closed submanifold of a Euclidean space of any dimension not just for (codimension-1) surfaces. Capturing the homotopy type of the complement in addition to that of the manifold substantially improves the quality of topological guarantee. For example, all closed curves have the same homotopy type (in fact are homeomorphic) and its is the homotopy type of the complement of the curve that distinguishes knots from one another. Similarly, a knotted torus is homeomorphic to an unknotted one while their complements have different homotopy types.

For uniform samples, the separation of critical points which is determined in terms of their distance from the manifold translates into a separation in terms of distance from the sample itself. In other words, if one sorts the critical points in the order of their distance to the sample, shallow critical points make a prefix of this ordering. Thus if one filters the flow complex by putting together the stable man-

*Duke University, Durham, NC 27708, sadri@cs.duke.edu

ifolds, i.e. cells associated to, critical points in all prefixes of this order, one is guaranteed to reach in this filtration a shape homotopy equivalent to the manifold in question.



As mentioned above, the union of stable manifolds of the remaining critical points then captures the homotopy type of the complement of the manifold. Since the filtration is regardless of the manifold, this statement is true for any manifold for which the given sample is a dense enough sample. For example, if the given sample is dense for a curve embedded on a torus and for the torus itself, the above filtration results homotopy equivalent reconstruction of both the curve and the torus, as well as their complements, in different stages.

2 Background and preliminaries

Let P be closed nonempty subset of $\mathbb{R}^n$. The complement of P is the open set $P^c = \mathbb{R}^n \setminus P$. For any point $x \in P^c$, let $h_P(x) = \inf_{y\in P} \|x - y\|$ be the *distance function* defined by P and let $A_P(x) = \{y \in P : \|x - y\| = h_P(x)\}$.

While the distance function h_P is not smooth, it induces a vector field v_P over P^c which behaves like the gradient of h_P in the sense that $v_P(x) \neq 0$ if and only if there is a unique direction of steepest ascent for h_P at x in which case the direction of this steepest ascent is given by $v_P(x)$ (See [10] for more general statement and details). The vector $v_P(x)$ at a point x is characterized by $v_P(x) = \frac{x-d_P(x)}{h_P(x)}$, where $d_P(x)$, called *driver of* x is the center of the smallest enclosing ball of $A_P(x)$, or equivalently, the closest point in $\operatorname{conv} A_P(x)$, the *convex hull* of $A_P(x)$, to x. The *critical points* of h_P are those points x for which $v_P(x) = 0$, or equivalently, $x = d_P(x) \in \operatorname{conv} A_P(x)$.

Lieutier [12] proved that if P^c is bounded, then Euler schemes defined by v_P on P^c uniformly converge and this results in a *flow map* $\phi_P : \mathbb{R}^+ \times P^c \to P^c$ (where $\mathbb{R}^+$ is the set of non-negative reals) which he also proved to be continuous (on both variables). Intuitively, $\phi_P(t, x)$ is the point y that is reached from following the vector field v_P for time interval of length t, starting at x, by infinitesimal movements proportional to the magnitude of v_P. The map ϕ_P has the classical properties of a flow map, namely $\phi_P(0, x) = x$, $\phi_P(s + t, x) = \phi_P(s, \phi_P(t, x))$, and for any point x and any $t \geq 0$, $v_P(\phi_P(t, x))$ is the *right-derivative* of $\phi_P(t, x)$. Lieutier also proved that h_P along any flow orbit, i.e. $t \mapsto h_P(\phi_P(t, x))$ is increasing and in addition satisfies

$$h_P(\phi_P(t, x)) = h_P(x) + \int_0^t \|v_P(\phi_P(\tau, x))\|^2 d\tau. \quad (1)$$

The special case where P is finite is of particular interest to us and the rest of this section goes over special properties of the flow maps in this case. Let $\operatorname{Vor} P$ and $\operatorname{Del} P$ respectively denote the Voronoi and Delaunay complexes induced by P. For any point $x \in \mathbb{R}^n$, we represent by $V_P(x)$ the *lowest dimensional face* of $\operatorname{Vor} P$ that contains x, and by $D_P(x)$ the face in $\operatorname{Del} P$ dual to $V_P(x)$. The set $A_P(x)$ is the vertex set of $D_P(x)$ and $d_P(x)$ becomes the closest point on $D_P(x)$ to x. It can be verified that all points in the relative interior of the same Voronoi face have the same driver. Since the affine hulls of a Voronoi face and its dual are orthogonal with total dimension n, they intersect in exactly one point. Thus if $V_P(x)$ and $D_P(x)$ intersect, then this intersection consists of a single critical point which is the driver of x. All critical points (except for the maximum at infinity) are characterized the same way (as intersection points of duals). Following [9], we make a *general position assumption* that all pairs of Voronoi and Delaunay objects that are dual to and intersect each other, do so in their relative interiors. The *index* of a critical point c is defined as the dimension of $D_P(c)$.

For a given flow map ϕ_P, the *flow orbit* of a regular point x, denoted $\phi_P(x)$ is defined as $\phi_P([0, +\infty), x)$. For a set T we use $\phi_P(T)$ for $\bigcup_{x\in T} \phi_P(x)$. Notice that by this definition $T \subseteq \phi_P(T)$.

For a critical point c of h_P, the set of all points x whose flow orbit converges to c is called the *stable manifold* of c and denoted by $\operatorname{Sm}(c) = \{x : \phi_P(+\infty, x) = c\}$. Although there is no flow out of a critical point c, we study the orbits of points very close to c. Some of these points flow into c while other flow away from it. We define the *unstable manifold* $\operatorname{Um}(c)$ of a critical point c, as the set of all points into which points arbitrarily close to c flow. Formally, $\operatorname{Um}(c) = \bigcap_{\varepsilon>0} \phi(B(c, \varepsilon))$, where $B(c, \varepsilon)$ denotes the open ball of radius ε centered at c. In other words, the unstable manifold of c consists of c and all the integral lines that start infinitesimally close to c.

Proposition 1 *Let P be finite. For a critical point c of h_P, $\operatorname{Um}(c) = \phi_P(V_P(c))$.*

A set T is said to be *flow-tight* for ϕ_P if $\phi_P(T) = T$. Stable and unstable manifolds of critical points and their union and intersections are flow tight. Let $\mathcal{C}_P$ be the set of critical points of h_P induced by P (including the critical point at infinity). The *(stable) flow complex* of P, denoted $\operatorname{Sfc} P$ is the collection of stable manifolds of all critical points in $\mathcal{C}_P$. Generically, the cell associated to an index k critical point is a topological open k-ball. Moreover, if for critical points $c, c' \in \mathcal{C}_P$, $c \in \partial \operatorname{Sm}(c')$, then $\operatorname{Sm}(c) \subset \partial \operatorname{Sm}(c')$.

Lemma 1 *If for $c \in \mathcal{C}_P$, $\operatorname{ind} c = k$, then every critical point $c' \in \partial\, Sm(c)$ has index less than k, provided that $Sm(c)$ does not intersect the $(n - k - 1)$-skeleton of*

Vor P. Under the same assumption, if $c \in \partial\, Um(c')$, *then* $\operatorname{ind} c' < \operatorname{ind} c$.

All but a measure-0 set of points P satisfy the requirement that $\mathrm{Sm}(c)$ must stay clear from faces of $\operatorname{Vor} P$ of dimension $n-k-1$ or smaller (See [14]).

By a *manifold* we refer to a C^2-smooth closed submanifold Σ of $\mathbb{R}^n$. The medial axis $M(\Sigma)$ of Σ consists of points in space with 2 or more closest points in Σ. The *reach* of Σ is the distance between Σ and $M(\Sigma)$. We assume the reach of Σ is strictly positive. Any point $x \notin M(\Sigma)$, has a unique closest point $\hat{x}$ in Σ. The half-line bounded at $\hat{x}$ through x hits $M(\Sigma)$ for the first time at a point $\check{x}$ (or at infinity).

A point set $P \subset \Sigma$ is a *uniform ξ-sample* of Σ if $\forall x \in \Sigma\ \exists p \in P : \|x-p\| \leq \xi$. For a given parameter $r \geq 0$, the union of balls $\bigcup_{p\in P} B(p,r)$ is denoted by $B^{(r)}(P)$. The *α-shape* of P of parameter r, denoted $K^{(r)}(P)$ is the underlying space of *restriction* of $\operatorname{Del} P$ to $B^{(r)}(P)$ (See [6]). The *flow shape* of P for parameter r, denoted $F^{(r)}(P)$ is the union of stable manifolds of critical points at distance $\leq r$ from P (See [4]).

3 Shallow versus deep critical points

For any point $x \in \mathbb{R}^n \setminus (\Sigma \cup M(\Sigma))$ let $\mu(x) = \|\check{x}-\hat{x}\|$. If $\check{x}$ is at infinity, then $\mu(x) = \infty$. Otherwise, the ratio $0 < \frac{\|x-\hat{x}\|}{\|\check{x}-\hat{x}\|} < 1$, is a relative measure of how close to Σ or $M(\Sigma)$ the point x is. It turns out [5, 3] that when a (possibly noisy) sample P of Σ satisfies some density requirements, then critical points of h_P are distributed, according to the above measure, into two distinguishable groups, one lying very close to Σ and the other to $M(\Sigma)$. We use a weaker version of the Lemma for uniform samples here.

Theorem 2 *Let P be an $\varepsilon\tau$-sample of a manifold Σ of reach τ with $\varepsilon \leq 1/\sqrt{3}$. Then for every critical point c of h_P, either $\|c-\hat{c}\| \leq \varepsilon^2\tau$, or $\|c-\hat{c}\| \geq (1-2\varepsilon^2)\tau$. In the former case we call c a* shallow *critical point and a in the latter case a* deep *one.*

Corollary 1 *Under the settings of Theorem 2, for every shallow critical point c of h_P, $h_P(c) \leq \sqrt{5/3}\cdot\varepsilon\tau$, and for every deep critical point c' of h_P, $h_P(c') \geq (1-2\varepsilon^2)\tau$.*

For any $0 \leq \delta < 1$, the *δ-tubular neighborhood* of a manifold Σ of reach τ is defined as the set $\Sigma_\delta = \{x \in \mathbb{R}^n : \|x-\hat{x}\| \leq \delta\tau\}$. Notice that $M(\Sigma) \subset \Sigma_\delta^c$.

Lemma 3 *For any $0 \leq \delta < 1$, $cl\,\Sigma_\delta^c$ is homotopy equivalent to Σ^c. In fact, the former is a strong deformation retract of the latter.*

Lemma 4 *Let P be an $\varepsilon\tau$-sample of a manifold Σ of reach τ with $\varepsilon \leq 1/(1+\sqrt{2})$. Then, $cl\,\Sigma_\delta^c$ is flow-tight under the flow ϕ_P, for any $\frac{\varepsilon^2}{1-\varepsilon} < \delta < 1-\varepsilon-\frac{\varepsilon^2}{1-\varepsilon}$. In particular this is true for $\delta = 1/2$.*

The above lemma implies that union of stable manifolds of shallow critical points is contained in Σ_δ for $\delta = \varepsilon^2/(1-\varepsilon)$ thus providing the Hausdorff distance guarantee for our reconstructions.

4 Homotopy type of the manifold

In this section we show that in a dense enough sample of a submanifold of $\mathbb{R}^n$, the union of stable manifolds of the shallow critical points has the same homotopy type as the manifold itself. This statement follows from the following sequence of results.

Lemma 5 [13] *Let Σ be a manifold of reach τ and let P be an $\varepsilon\tau$-sample of Σ for any $\varepsilon \leq \frac{1}{2}\sqrt{3/5}$. Then $B^{(r)}(P)$ deformation retracts (and is in particular homotopy equivalent) to Σ, for any $2\varepsilon\tau < r < \sqrt{3/5}\cdot\tau$.*

Lemma 6 [6] *For any $r \geq 0$, $B^{(r)}(P)$ and the α-shape $K^{(r)}(P)$ are homotopy equivalent.*

Lemma 7 [4, 1] *For any r, the flow shape $F^{(r)}(P)$ and the α-shapes $K^{(r)}(P)$ are homotopy equivalent.*

Theorem 8 *Let Σ be a manifold of reach τ and let P be an $\varepsilon\tau$-sample of Σ for $\varepsilon \leq \frac{1}{2}\sqrt{3/5}$. Then Σ is homotopy equivalent to the union U of stable manifolds of shallow critical points of h_P.*

Proof. For a critical point c of h_P, by Corollary 1 $h_P(c) \leq \sqrt{5/3}\cdot\varepsilon\tau$ if c is shallow and $h_P(c) \geq (1-2\varepsilon^2)\tau$ if c is deep. For $\varepsilon < \frac{1}{2}\sqrt{3/5}$ the latter bound is strictly greater than the former and therefore there is a positive value r for which $h_P(c) < r$ for every shallow critical point c and $h_P(c') > r$ for every deep critical point c'. Thus the flow shape $F^{(r)}(P)$ is precisely the union of stable manifolds of shallow critical points of h_P with respect to Σ. Lemmas 5, 6, 7 now imply that this union is homotopy equivalent to Σ. □

5 Homotopy type of the complement of the manifold

In this section we prove that the union of stable manifolds of deep critical points has the homotopy type of Σ^c using the continuity of the flow map ϕ_P. The technique is inspired from the work of Lieutier [12]. A proof can found in [14].

Theorem 9 *Let P be a finite set of points in $\mathbb{R}^n$. If for sets $Y \subset X \subset \mathbb{R}^n$, X and Y are both flow-tight for ϕ_P, i.e. $\phi_P(X) = X$ and $\phi_P(Y) = Y$, and if $X \setminus Y$ is bounded, and, finally, if there is a constant $c > 0$ for which $\|v_P(x)\| \geq c$ for all $x \in X \setminus Y$, then X and Y are homotopy equivalent.*

A difficulty in using the above theorem is that ϕ_P is proven in [12] to be continuous on P^c as long as it is a *bounded* set. This can be overcome by clipping the space with a very large ball, thus letting $P_0 = P \cup B^c$ where B is a very large ball satisfying $P \subset \frac{1}{5}B$. It can then be verified that within $\frac{1}{2}B$, ϕ_P and ϕ_{P_0} agree which is enough for what we want to prove. In the sequel $\mathcal{C}_\Sigma$ denotes the set of shallow critical points of P where P is an $\varepsilon\tau$-sample of a manifold σ of reach τ. The value of ε is determined later. For shorthand, we write S for Σ^c as well S_δ for Σ^c_δ.

Lemma 10 *Let c be a critical point of h_P and let $U \subseteq \mathbb{R}^n$ be a flow-tight set for ϕ_P with $c \notin U$. Let $V = \operatorname{rel\,int} V_P(c)$. For $r \geq 0$, let $V_r = V \cap B(c,r)$. Then for every $r \geq 0$, U and $U \setminus V_r$ have the same homotopy type and if $U \cap B(c,r) \subset V$ then $U \setminus V_r$ is flow-tight for ϕ_P.*

Theorem 11 *Let $\varepsilon \leq \frac{1}{2}\sqrt{3/5}$. Let $\tilde{S} = \bigcup_{c \in \mathcal{C} \setminus \mathcal{C}_\Sigma} Sm(c)$ be the union of stable manifolds of all deep critical points of h_P with respect to Σ. Let $U_\Sigma = \bigcup_{c \in \mathcal{C}_\Sigma} Um(c)$ be the union of unstable manifolds of all shallow critical points. Then $\tilde{S}$ is homotopy equivalent to S.*

Proof. (*sketch*) We use Theorem 9 to show that $X = S$ is homotopy equivalent to $Y = \tilde{S}_{1/2}$ which is itself homotopy equivalent to S by Lemma 3. The main difficulty in the proof is that although both X and Y are flow-tight for ϕ_P, $\|v_P\|$ can be arbitrarily small in $X \setminus Y$ because the boundary of X can contain critical points that drive points in the interior of X arbitrarily close to them. To handle this difficulty, a first idea is to use Lemma 10 to remove from X a neighborhood of these critical points, thus creating a strictly positive distance between these critical points and points in the trimmed X. However, in order to do this in a manner that ensures the trimmed X is still flow tight, this has to be done in several steps where the i-th step gets rid of critical points of index i. We thus first delete from X a neighborhood of every critical point of index-0 to get a flow-tight set X_0 that by Theorem 9 will be homotopy equivalent to the set $\tilde{X}_0$ consisting of the union of Y and the *unstable* manifolds of critical points of index 1 and higher on boundary of X, restricted to X. One can then remove from x a neighborhood of all index-1 critical points resulting a set X_1 that using Lemmas 1 and 10 is flow-tight for ϕ_P. Applying Theorem 9 then results a set $\tilde{X}_1$ consisting of the union of Y and unstable manifolds of critical points of index 2 or higher clipped by X. Continuing this way, all critical points on the boundary of X can be eliminated resulting a sequence of homotopy equivalent shapes $X_0, \tilde{X}_0, X_1, \tilde{X}_1, \ldots, X_n, \tilde{X}_n$ the first of which is homotopy equivalent to X and the last one to Y. □

Corollary 2 *Let $\Sigma_1, \ldots, \Sigma_s$ be manifolds of various dimensions for all of which the same sample P is an $\varepsilon\tau_i$-sample where τ_i is the reach of Σ_i, $i = 1, \ldots, s$. If $c_1, \ldots, c_m$ are the set of critical points of h_P sorted such that $h_P(c_1) < \cdots < h_P(c_m)$, then for each i, there is a j_i such that $\bigcup_{j \leq j_i} Sm(c_j) \simeq \Sigma_i$ and $\bigcup_{j > j_i} Sm(c_j) \simeq \Sigma_i^c$.*

References

[1] K. Buchin and J. Giesen. Flow complex: General structure and algorithms. In *Proc. 16th Canad. Conf. Comput. Geom.*, pages 270–273, 2005.

[2] F. Cazals and J. Giesen. *Delaunay Triangulation Based Surface Reconstruction*, chapter in Effective Computational Geometry of Curves and Surfaces, Jean-Daniel Boissonnat and Monique Teillaud (Editors). Springer-Verlag, 2006.

[3] F. Chazal and A. Lieutier. Topology guaranteeing manifold reconstruction using distance function to noisy data. In *Proc. 22nd Annu. ACM Sympos. Comput. Geom.*, pages 112–118, 2006.

[4] T. K. Dey, J. Giesen, and M. John. Alpha-shapes and flow shapes are homotopy equivalent. In *Proc. 35th Annu. ACM Sympos. Theory Comput.*, pages 493–502, 2003.

[5] T. K. Dey, J. Giesen, E. A. Ramos, and B. Sadri. Critical points of the distance to an epsilon-sampling of a surface and flow-complex-based surface reconstruction. In *Proc. 21st Annu. ACM Sympos. Comput. Geom.*, pages 218–227, 2005.

[6] H. Edelsbrunner. The union of balls and its dual shape. *Discrete Comput. Geom.*, 13:415–440, 1995.

[7] H. Edelsbrunner. Surface reconstruction by wrapping finite point sets in space. *Discrete & Computational Geometry*, 32:231–244, 2004.

[8] J. Giesen and M. John. Surface reconstruction based on a dynamical system. *Computer Graphics Forume*, 21:363–371, 2002.

[9] J. Giesen and M. John. The flow complex: A data structure for geometric modeling. In *Proc. 14th ACM-SIAM Sympos. Discrete Algorithms*, pages 285–294, 2003.

[10] K. Grove. Critical point theory for distance functions. *Symposia in Pure Mathematics*, 54(3):357–385, 1993.

[11] A. Hatcher. *Algebraic Topology.* Cambridge University Press, 2001.

[12] A. Lieutier. Any bounded open subset of $\mathbb{R}^n$ has the same homotopy type as its medial axis. *Computer-Aided Design*, 36(11):1029–1046, 2004.

[13] P. Niyogi, S. Smale, and S. Weinberger. Finding the homology of submanifolds with high confidence from random samples. *Discrete & Computational Geometry*, 2006, to appear.

[14] B. Sadri. *Surface and Medial Axis Topology Through Distance Flows Induced by Discrete Samples.* PhD thesis, University of Illinois at Urbana-Champaign, 2006.

Surface Deformation on a Discrete Model for a CAD System

Ioana G. Ciuciu* Frederic Danesi† Yvon Gardan‡ Estelle Perrin§

Abstract

Discrete geometry is becoming one of the most powerful and easy to use fields of Mathematics which applies with great success in most of the Computer Science areas concerned with the geometric modelling and with the visualisation of shapes. We aim to obtain a surface modelling tool within a larger CAD framework, the DIJA system developed by our research team, to allow local deformation of discrete surfaces in a manner as free as the constraints specific to a certain trade would allow.

1 Introduction

Classical Computer Aided Design systems for surface-modelling are built on continuous models such as polynomial surfaces and super quadrics. The drawback with these systems is that they are complex in terms of representation, manipulation and visualisation of surfaces, difficult to apply and time consuming [1, 2]. The deformations that we can operate on these models are often limited to global deformations and the deformed shapes are restricted to a certain topology [3, 4, 5].

Opposite to these methods lie the discrete geometric modelling techniques which provide the simplest way to represent the geometry of any surface and to make direct interactions on this one [6, 7, 8]. We can though manipulate surfaces with any kind of shape in the simplest and fastest manner. The representations are point-based and generally constitute discretizations of the continuous shapes. This greatly facilitates the internal storing and processing of the model on a workstation. The user can interact directly with the points lying on a surface deciding where a point should go and which should be the region of influence of the displacement [4].

*ERT Gaspard Monge/CReSTIC/IFTS, Charleville-Mezieres, France, University of Reims Champagne-Ardenne, oana.ciuci@gmail.com

†DINCCS/ERT Gaspard Monge/CReSTIC/IFTS, Charleville-Mezieres, France, University of Reims Champagne-Ardenne, frederic.danesi@dinccs.com

‡ERT Gaspard Monge/CReSTIC/IFTS, Charleville-Mezieres, France, University of Reims Champagne-Ardenne, gardan@infonie.fr

§ERT Gaspard Monge/CReSTIC/IFTS, Charleville-Mezieres, France, University of Reims Champagne-Ardenne, estelle.perrin@univ-reims.fr

In this paper we propose a discrete surface modelling tool to allow local deformations on surfaces in the domain of CAD. The deformations are submitted to geometric constraints and obey laws specific to a certain trade specified by the user at the beginning of the modelling. This approach is part of a much larger CAD framework developed by our research fellows, the DIJA project. DIJA is a federative concept whose idea is to create a functional, distributive, collaborative and intuitive CAD system. Only the functional and intuitive aspects count here.

The intuitiveness is reached by the bias of trade-based tools which help the user emerge into his/her world and gives him/her the possibility to make geometric transformations only by using his/her know-how. For example, a tool for blacksmiths would be to soften the filets of their shapes by applying a logarithmic function in order to improve metal flaw. This definition of tool must not be related to tools using peripheral devices [9]. This fact is not convenient for us, as we want to develop tools which conserve the classical peripheral device: the mouse.

The functional aspect is reached by a progressive decomposition of the object, which is at the opposed pole from the modelling used by the actual CAD systems [10]. The precision of the resulting form is achieved thanks to deformations which are no longer done on a classical model but on some visible entities called dialog elements. Dialog elements are elements representing the objects silhouette and which permit to obtain visual details on the modelised object.

Actually, only deformations applied on 2D dialog elements are implemented and we are trying to improve them in order to perform them on a particular 3D dialog element, the surface under a supplementary constraint that is to obtain satisfactory response times.

In this article we propose a new local deformation method of surfaces in order to preserve constraints induced by fillets. For reaching our purposes we choose a discrete model which fits the best the synthetic modelling approach and the constraints imposed by a specific trade. In section 2 we present a state of the art on the discrete deformation models highlighting and motivating the main directions of our research study. Section 3 presents the geometric model which is at the basis of our system. In section 4 we explain our first practical results and we finalize by resuming this paper and by making our future intentions known.

2 Overview

In this section we present an overview on the different surface deformation techniques with the focus on those which interest us the most, keeping in mind the fact that we want a modelling method which applies deformations on the 3D face dialog element using trade-based intuitive tools with acceptable response times from the system.

The first studies in the field of surface deformation are made using the explicit deformation technique proposed by Barr [11]. Other approaches use the mechanical properties of solids for expressing the deformation as a system which changes in time, dependent on inertial and elastic properties. These methods do not correspond to our needs because they are time consuming and not necessarily intuitive. We concentrate on purely geometric spatial deformations which give the geometric form of the physical result, the model not being though induced by physical properties.

Other methods are based on the geometric description of the object [12]. In this case, the user must know the geometric properties of the model in order to make changes on this one, which is non-intuitive.

Free Form Deformation - introduced by Sederberg and Parry in [4] - is a technique which can be applied to any object, independently of its geometrical and topological description. The model is based upon a patch of control points placed along three axes mutually perpendicular which define a tri-variate Bezier volume. This model has been extended to models where the control patch has an arbitrary topology. In [13], Coquillart proposes a model which allows combining the control meshes. Lazarus [4] introduced the axial-based deformation in which a deformation axis is associated with the object and the deformation of the last one is made in accordance to that applied to the axis. The major inconvenient of the patch-based or axial-based deformations is the lack of intuitiveness due to inexact control over the surface and the impossibility to apply local deformations to it. Borrel and Bechmann [14] have tried to eliminate this problem by proposing a constraint-based method. It basically consists in fixing displacement constraints on the points of an object and ensuring their satisfaction through the deformation. The resulting object is a combination of deformation functions called decay functions and the imposed constraints. Apart from the displacement constraint and from the type of decay function, the user can specify also the extent of the constraints' influence on the surface.

Decay functions have great importance over the deformation process, since they apply a mathematical function directly and locally on the points of the surface. They establish the deformation style, the action to be taken on the surface and the radius of influence. The style gives the visual aspect of the deformation, the action is that of locally deforming the surface and the radius of influence specifies the number of neighbour vertices affected. In [7] we are presented a series of deformation styles, such as Goo, Bell, Cusp, Cone, and Flat.

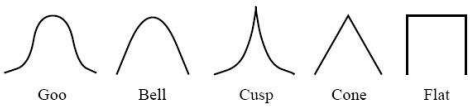


Figure 1: Shapes of decay functions [7].

The inconvenient with the constrained based deformations using decay lies in the way of linking the local deformed region of the surface with the rest of the surface.

An interesting solution in this direction would be the subdivision surfaces which can be used in order to smooth the deformed surface. This gives visually pleasant results but without respecting the trade-based constraints.

The next paragraph reminds us how deformations are made under the DIJA system and we finish by studying how we could solve the trade-based constraints problem induced by the constrained-based method and decay functions.

3 DIJA deformations

The architecture of the DIJA system is based upon five abstraction levels, each of which being different from the others in terms of possible tasks and vocabulary.

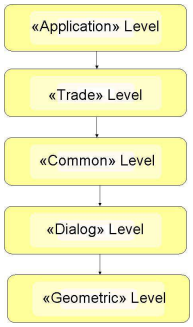


Figure 2: The different abstraction levels.

Deformations are the heart of the "synthetic approach" and are possible at the "trade", "common" and "dialog" levels thanks to the different available tools. They are based on visual elements that can be found at the "dialog" level. There exist three types of dialog elements: the characteristic line and contour, the fiber and the face. The characteristic line and contour represent visible lines on the object's surface, the

fibre is a line giving the general silhouette of the object. Those are 2D objects. The results of a deformation process using these elements is seen in Fig.3. The term face corresponds to a 3D element and designs a homogeneous surface or a portion of a homogeneous surface.

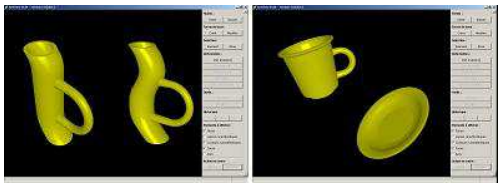


Figure 3: Objects modelised with 2D dialog elements.

Deformations are done on any geometric model - either a B-Rep model, a CSG, an octree, a mixed model or a model from the research field - stored at the "geometric" level, provided that this one can answer the demands of the "dialog" level and in particularly collect the points of the face element starting from these models.

With the 3D deformation method we wish the user to be able to deform the surface in the following manner: he/she starts with selecting a point on the surface, then he/she indicates a radius of influence encircling the selected point and a decay function for the shape of the deformation. Then he/she pulls the selected point out of the surface and it modifies in accordance to his/her users choices. In the following section we present the deformation model we created in order to conserve the trade-based constraints.

4 Implementation

Our CAD system is conceived to fit several trades, each of them having its own set of trade-specific deformation tools which apply on an initial surface in order to deform it. The user starts his/her modelling process by selecting a trade, an initial shape, and by specifying the parameters which correspond to the trade rule to be applied. It is then the system's task to interpret the user choices and to find the parameterized deformations to apply to the surface in a way that ensures its quality in terms of continuity.

The application was developed using the Java programming language and the VTK visualization library. It was tested on a 2.4GHz duo processor with 2GB of memory.

The internal model is formed by 3D points and triangles read from an STL source file. The surface we use is planar and contains a list of points, i.e. all the points on the surface at a deformation level, a list of triangles formed by these points and a value specifying the radius of influence of the deformation at each step. Its internal structure changes at each deformation step. The surface has a deformation entity associated, which indicates at any time the shape of the applied deformation, and a smoothing entity which specifies the type of smoothing operation applied.

The deformation model contains three types of deformation corresponding to different deformation laws: cone, bell, and cylinder. We have chosen them for their simplicity in order to simulate the basic deformations a CAD user can perform to deform the object. Examples of the three are presented in Fig.4.

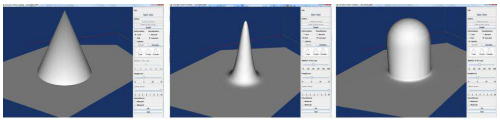


Figure 4: The three types of deformation allowed: cone, bell, cylinder.

The three laws increase progressively in complexity and quality. If the first law simply pushes a point out of the surface in a cone-like deformation with no worry for the continuity of the resulted surface, the second one increases the quality of the deformation by using the Gauss function which assures itself the continuity, while in the third case we had to assure ourselves the smoothness between the deformed region and the rest of the surface by applying an appropriate mathematical function. We used 1/x at different scales on the interval [0.25, 12] on the points inside the region of influence. The scaling of the cylinder basis is done interactively with a slider. Three examples are done in Fig.5. The first one presents the cylinder in its original shape, while the other two use a version of 1/x at a much larger scale to represent it.

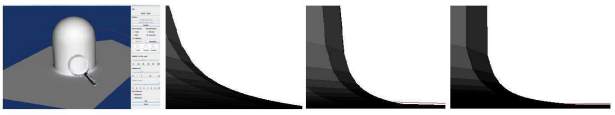


Figure 5: Improving the smoothness by scaling the cylinder basis.

Fig.6 shows an example of a deformation process applied on a plane. The user starts by marking on this one an initial point and a radius of influence which in our case determines a disc of influence. Then he/she chooses the bell deformation law and indicates a displacement position in the 3D space by a mouse left-click. The last two pictures show the results after having modified the displacement position and the radius of influence respectively.

5 Conclusion

We created a discrete geometric model which allows the representation and deformation of surfaces in the domain of CAD. The representation is realized in

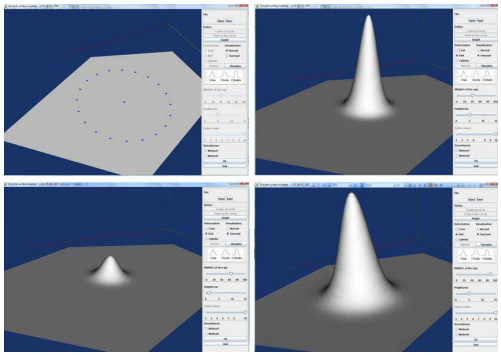


Figure 6: Bell deformation applied on a planar surface. Upper left: the starting surface with the chosen point and area of deformation; upper right: the result of a bell deformation; lower left: the result of modifying the displacement point; lower right: the surface after having modified the area of influence

terms of triangular polygonal meshes, while the deformation model is based on constrained deformations and decay functions. In practice, this model is concretized in a prototype which permits interactive real-time deformations locally on planar 3D discrete surfaces according to the choices made by the user. In present we concentrate on the deformation of free form discrete surfaces following from laws straight related to real world rules applied in various trades and upon the development of virtual tools.

6 Future work

The prototype we have developed will be further transformed into a real CAD application in which the deformation laws will be generalised and strongly correlated to trade rules from different domains. We will have to find a way to associate any trade-rule with any type of deformation to ensure generality. For so doing we need to extend our internal model to contain vicinity relations between vertices, edges and triangles. The deformations will be done on surfaces of any form by making use of the new implemented intuitive virtual tools. We will also implement a method to redescend the results of the deformation at the "geometric" level, this one not being done purely at a discrete geometric level beacuse the parameters of the tool - forme, distance, etc. - are known.

References

[1] F. Danesi C Dartigues Y. Gardan E. Perrin. *World wide web adapted geometric model in the context of functional design.* 4th International Conference on Design of Information Infrastructure Systems for Manufacturing, 2000.

[2] A. Johnson A. Thornton C. Fong. *Modelling functionality in cad: implications for product representations.* 9th International Conference on Engineering Design, 1993.

[3] H. Masuda R. Ohbuchi. *Coding Topological Structure of 3D CAD Models.* Computer Aided Design, 2000.

[4] T. Sederberg S. Parry. *Free-form Deformation of Solid Geometric Models.* Computer Graphics, 1986, Louisina, USA.

[5] F. Lazarus S. Coquillart P. Jancene. *Axial deformations: an intuitive deformation technique.* Computer Aided Design, 1994.

[6] J. Montagnat H. Delingette N. Ayache. *A review of deformable surfaces: topology, geometry and deformation.* Image and Vision Computing, 2001.

[7] J. Bill S. Lodha. *Computer Sculpting of Polygonal Meshes Using Virtual Tools.* Tech.Rep.UCSC-CRL-94-27, Baskin Center For Computer Engineering and Information Sciences, 1994, California, USA.

[8] R. Parent. *A System For Sculpting 3D Data.* Computer Graphics, 1977.

[9] J. Gargus B. Kim I. Llamas J. Rossignac C. Shaw. *Finger Sculpting with Digital Clay.* Technical Report GIT-GVU-02-22, 2002.

[10] F. Danesi L. Denis Y. Gardan E. Perrin. *Basic Components of the DIJA project.* The 7th ACM Symposium on Solid Modeling and Applications 2002 (SM02) Saarbrcken Germany 17-21 juin 2002.

[11] A. Barr. *Global and Local Deformations of Solid Primitives.* Computer Graphics, 1984.

[12] J. Blinn. *A Generalization of Algebraic Surface Drawing.* ACM Transactions on Graphics, 1982.

[13] S. Coquillart. *Extended Free-Form Deformation: a sculpting tool for 3D geometric modelling.* International Conference on Computer Graphics and Interactive Techniques, 1990.

[14] P. Borrel D. Bechmann. *Deformation of n-dimensional objects.* Comput.Geometry Appl., 1991.

[15] W. Hsu J. Hughes H. Kaufman. *Direct Manipulation of Free-Form Deformations.* International Conference on Computer Graphics and Interactive Techniques, 1992.

[16] J. Allan B. Wyvill I. Witten. *A Methodology for Direct Manipulation of Polygon Meshes.* New Advances in Computer Graphics, 1989.

Optimal Insertion of a Segment Highway in a City Metric

Matias Korman* Takeshi Tokuyama†

Abstract

Given two sets of points in the plane, we are interested in locating a highway h of length ℓ such that an objective function on the *city distance* between points of the two sets is minimized (where the city distance is measured with speed $v > 1$ on h and 1 in the underlying metric elsewhere). Extending the results of Ahn et al. ([6]), we consider the option that there are some already built roads. Our algorithm has polynomial time complexity, and unified structure for several optimization criteria.

Keywords: facility location, city metric, computational geometry, optimization problems, algorithms, urban planning.

1 Introduction

We consider the following problem: Given two sets of points in the plane $\mathcal{S}$ and $\mathcal{T}$ (called *sources* and *sinks* respectively) and a set of isothetic (i.e: horizontal or vertical) segments $\mathcal{H}$ called *highways*, we locate another highway h that minimizes an objective function related to the *city distance* between sources and sinks considering that we can travel v times faster on a highway. This problem arises naturally in urban planning where source points correspond to houses, sinks to working centers, and highways to real highways or railways.

Abellanas *et al.* ([5]) considered the construction of the Voronoi diagram for point sets under the L_1 metric given an isothetic and monotone highway of speed $v > 1$. Aichholzer *et al.* ([3]) introduced efficient algorithms for calculating Voronoi diagrams under a set of isothetic highways. The problem of constructing a line highway to minimize the maximum distance was considered by Cardinal and Langerman ([6]) in the context of a facility location problem. Ahn *et al.* ([1]) and afterwards Cardinal *et al.* ([7]) further improved those results and considered different highway types. Up to our knowledge, there has been no research in optimal highway location that considers previously built highways or focuses in an objective function other than maximum travel distance. In this paper, we introduce a general algorithm to build a segment highway of length ℓ (a fixed value) that can be adapted to work for different evaluation functions under the L_1 metric. In Figure 1 we can see a sample problem with four sources (filled points) each one connected to its closest sink (hollow points).

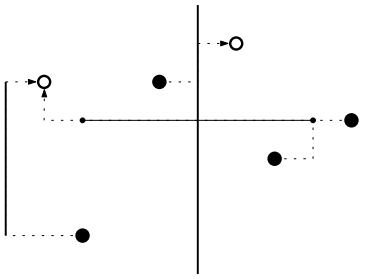


Figure 1: Sources and their paths to their closest sinks

2 Definitions and results

Given the sets of sources $\mathcal{S}$, sinks $\mathcal{T}$, and highways $\mathcal{H}$, we call the triplet $(\mathcal{S}, \mathcal{T}, \mathcal{H})$ a *transportation configuration*. Other than a configuration, an underlying metric d and a constant *speed* $v > 1$ are needed to define the problem. We consider that each highway in $\mathcal{H}$ is either one of the following types: *railway* (we can only enter/exit at endpoints) or *road* (one can enter/exit at any point of the highway). The travel time of the path $\pi = (s = p_0, \ldots, p_k = t)$ is defined as $|\pi| = \sum_{0 \le i < k} \frac{1}{v_i} d(p_i, p_{i+1})$, where v_i is v if the segment p_i, p_{i+1} is a railway or is on a road, 1 otherwise. The transportation distance between s and t is defined by $d_{\mathcal{H}}(s,t) = \inf_{\pi \in \mathcal{P}(s,t)} |\pi|$, where $\mathcal{P}(s,t)$ is the set of all s-t-paths. Since we consider the L_1 distance as d, we can observe that any path between s and t is a sequence of isothetic segments (p_i, p_{i+1}).

For *each* source s in $\mathcal{S}$ we consider a function $cost(s) = \odot_{t \in \mathcal{T}} d_{\mathcal{H} \bigcup \{h\}}(s,t)$, where $\odot$ stands for an operation in $\{\sum, \min, \max\}$, and $d_{\mathcal{H} \bigcup \{h\}}(s,t)$ is the city distance with the set of highways $\mathcal{H} \bigcup \{h\}$. Each function $cost(s)$ indicates a statistical measure of how far the sinks are from source s. Our objective function is $\Phi = \odot'_{s \in \mathcal{S}} cost(s)$, which shows an aggregated measure of $cost(s)$ for $s \in \mathcal{S}$ using an operation $\odot' \in \{\sum, \min, \max\}$.

Main result. The main result of this paper is the introduction of a new algorithm to find the optimal location of a new segment highway in a transportation

*Graduate School of Information Sciences (GSIS), Tohoku University, mati@dais.is.tohoku.ac.jp

†Graduate School of Information Sciences (GSIS), Tohoku University, tokuyama@dais.is.tohoku.ac.jp

$\odot$ / $\odot'$	Road			Railway		
	min	max	$\sum$	min	max	$\sum$
min	$O(STH^3)$	$O^*(ST^2H^6)$	$O(ST^2H^6)$	$O(STH^2)$	$O^*(ST^2H^4)$	$O(ST^2H^4)$
max	$O^*(S^2T^4H^{12})$	$O^*(S^2T^2H^6)$	$O^*(S^2T^4H^{12})$	$O^*(S^2T^4H^8)$	$O^*(S^2T^2H^4)$	$O^*(S^2T^4H^8)$
$\sum$	$O^*(S^2T^4H^{12})$	$O^*(S^2T^4H^{12})$	$O(S^2T^2H^6)$	$O^*(S^2T^4H^8)$	$O^*(S^2T^4H^8)$	$O(S^2T^2H^4)$

Table 1: Computational cost of locating a segment highway under the evaluation function $\Phi = \odot' \odot d_{\mathcal{H}(s,t)\bigcup\{h\}}$

configuration with already built highways. Although the algorithm presents a unified approach (independent of the objective function), the exact computational cost depends on $\mathcal{S}$, $\mathcal{T}$ and $\mathcal{H}$ as well as the particular evaluation function. The results are summarized in Table 1 (for clarity reasons we denote $S = |\mathcal{S}|$, $T = |\mathcal{T}|$, and $H = |\mathcal{H}|$).

		$\odot$		
		min	max	$\sum$
$\odot'$	min	$O^*(kST)$	$O^*(k^2ST^2)$	$O^*(k^2ST^2)$
	max	$O^*(k^4S^2T^4)$	$O^*(k^2S^2T^2)$	$O^*(k^4S^2T^4)$
	$\sum$	$O^*(k^4S^2T^4)$	$O^*(k^4S^2T^4)$	$O(k^2S^2T^2)$

Table 2: Cost of finding the minimum of $f = \odot'_i \odot_j f_{i,j}$

3 Overview of the algorithm

The following is our base problem: building a vertical segment highway h in a transportation network with a single source s and a single sink t. The insertion of h is parameterized by the location $\beta = (\xi, \eta)$ of its bottom endpoint. We may denote by h_β for the inserted highway and also denote $\mathcal{H}_\beta$ or $\mathcal{H}_{\xi,\eta}$ for the updated set of highways (i.e: $\mathcal{H}_\beta = \mathcal{H}_{\xi,\eta} = \mathcal{H}\bigcup\{h_\beta\}$). Regarding as a function of parameters $x = \xi$ and $y = \eta$, we define $f_{s,t}(x, y) = d_{\mathcal{H}_{x,y}}(s, t)$. In a general situation where we have more than one source and sink, our objective function Φ is described by using the set $\{f_{s,t} | s \in \mathcal{S}, t \in \mathcal{T}\}$. We now proceed to give an informal explanation of the algorithms: compute $f_{s,t}$ function for every possible pair source-sink, and then merge those functions to obtain the minimum of Φ.

We say that the complexity of a piecewise linear function f is k if there exists a subdivision of the domain of f (whose complexity is k) such that f restricted to each cell of the division is linear. The following lemma enables us to solve the problem in a unified fashion irrespective to the choice of our objective function:

Lemma 1 *For any $s \in \mathcal{S}$, $t \in \mathcal{T}$ we have that $f_{s,t}(x, y)$ is a piecewise linear function in the two coordinates of the new highway. Moreover, given $\mathcal{F} = \{f_{i,j}(x, y) | 1 \le i \le S, 1 \le j \le T\}$, a family of piecewise linear two dimensional functions with same domain and complexity k each, the function $f(x, y) = \odot'_i \odot_j f_{i,j}(x, y)$ is a piecewise linear function (where $\odot, \odot' \in \{\sum, \min, \max\}$). Moreover, the point $p = (x, y)$ minimizing $f(x, y)$ can be found in polynomial time as shown in Table 2.*

Proof of the first part of the lemma comes from linearity of L_1 distance, while the latter part comes from the general theory of the lower envelope of bivariate piecewise linear functions ([8]). For simplicity, polynomial factors of the inverse Ackermann function have been ignored (O^* notation is used wherever needed). That is, $k \in O^*(f(x)) \leftrightarrow k \in O(f(x)\alpha(x))$, where $\alpha(x)$ is the inverse of the Ackermann function.

4 Computing $f_{s,t}(x, y)$ function for a fixed source and sink

4.1 Preliminaries

According to the overview, we will focus on computing $f_{s,t}(x, y)$ function for a fixed a source s and sink t. We define the shortest path map [4] that is defined as follows in our context:

Given a point p and $\mathcal{H}$, we define $SPM(p, \mathcal{H}, \delta) = \left\{q \in \mathbb{R}^2 | d_{\mathcal{H}}(p, q) = \delta\right\}$, which is a polygon. Let $V(p, \mathcal{H}, \delta)$ be the set of vertices of $SPM(p, \mathcal{H}, \delta)$. The straight skeleton of p is defined as $SK(p, \mathcal{H}) = \left\{q \in \mathbb{R}^2 | \exists \delta \ge 0, q \in V(p, \mathcal{H}, \delta)\right\}$. Let $SPM(s, \mathcal{H})$ be the division of the plane into the regions defined by $SK(p, \mathcal{H})$. The shortest path map divides the plane into regions such that points in the same region have topologically identical shortest paths to p.

The following theorem is given by Bae et al. ([2]):

Theorem 2 *$SPM(p, \mathcal{H})$ can be constructed in $O(H \log H)$ time and stored in $O(H)$ space for any given set of isothetic segment highways $\mathcal{H}$ and $p \in \mathbb{R}^2$.*

We characterize the usage of roads in shortest paths. Given a point p, we define the North neighboring road as the first road that we find along the vertical semiline $\{p + (0, t) | t \ge 0\}$ if it exists. The point found in the intersection is called its North neighbor point p^N (analogously we define p^S, p^E and p^W). Let $N(p)$ the set of neighboring points of p, V the set of

endpoints of highways in $\mathcal{H}$, and I the set of intersection points of roads. Given a point p, we define $M_p(\mathcal{H}) = V \bigcup I \bigcup N(p) \bigcup_{q \in V} N(q)$.

A path $\pi = (s = p_0, \ldots, p_k = t)$ is called *primitive* if it has at most 1 bend, passes through at most one highway and does not pass through any point in $M_t(\mathcal{H})$. A primitive path with shortest length is called a *shortest* primitive path. The following lemma is given in [2], and implies that even if we allow hopping on and off of a road r at any point, there is a finite amount of such points in a shortest path:

Lemma 3 *For any $s \in \mathcal{S}$ and $t \in \mathcal{T}$ in a transportation configuration, there exists a shortest path π from s to t such that π is a sequence of primitive paths whose endpoints (except s) are in $M_t(\mathcal{H})$.*

4.2 Computing $f_{s,t}(x, y)$ from the shortest path map

We compute $f_{s,t}(x, y)$ from $SPM(s, \mathcal{H})$ and $SPM(t, \mathcal{H})$. If h is a road, For a fixed sink t and a point β (corresponding a potential location of the new highway h) we define $E_t(\beta)$ as the set of points in $M_t(\mathcal{H}_\beta)$ that lie in h_β (i.e: $E_t(\beta) = M_t(\mathcal{H}_\beta) \bigcap h_\beta$). Intuitively, $E_t(\beta)$ is the set of candidates of the entrance or exit point from h_β in the shortest path from s to t (see Figure 2). If h is a railway, we define $E_t(\beta) = \{\beta, \beta + (0, \ell)\}$, since the only possible exit/entry points to h are endpoints of h.

Let $k(\beta) = |E_t(\beta)|$. For any $i \in [1, k(\beta)]$, let β_i be the point in $E_t(\beta)$ whose y coordinate is the $i - th$ lowest one. Note that $\beta_1 = \beta$ and $\beta_{k(\beta)} = \beta + (0, \ell)$.

For $i, j \in [1, k(\beta)]$, the length of the shortest s-t path entering h_β at point β_i and exiting at point β_j equals

$$g_{i,j}(\beta) = d_{\mathcal{H}}(s, \beta_i) + d(\beta_i, \beta_j)/v + d_{\mathcal{H}}(\beta_j, t).$$

We define

$$f^{new}_{s,t}(\xi, \eta) = \min_{i,j \in [1,k(\beta)]} g_{i,j}(\beta).$$

By definition, for any $s \in \mathcal{S}$, $t \in \mathcal{T}$ and $\mathcal{H}$, we have

$$f_{s,t}(x, y) = \min\{d_{\mathcal{H}}(s, t), f^{new}_{s,t}(x, y)\}.$$

Given a region R, we consider the translation of R vertically by ℓ, and denote it by

$$R^\ell = R - (0, \ell) = \{(x, y) \in \mathbb{R}^2 | (x, y + \ell) \in R\}.$$

Accordingly, we define $\mathcal{R}^\ell = \{R^\ell_1, \ldots R^\ell_k\}$ for a plane subdivision $\mathcal{R} = \{R_1, \ldots R_k\}$.

We can observe that for the majority of locations for β, the set $E_t(\beta)$ is stable against a small movement of β (except for its top and bottom points). We will thus find a subdivision $\mathcal{R} = \mathcal{A}(t, \mathcal{H})$ of the plane

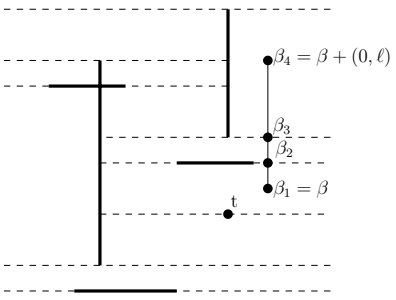


Figure 2: $\mathcal{H}$ (bold), its extension $T(\mathcal{H}, t)$ (dashed), and exit points $E_t(\beta)$ of a highway

such that, in every cell $c \in \mathcal{A}(t, \mathcal{H})$, the y-value of β_i is independent of choice of the point $\beta = (\xi, \eta) \in c$ if $i \neq 1, k(\beta)$. The subdivision $\mathcal{A}(t, \mathcal{H})$ is defined as follows: we draw horizontal rays to both sides from each endpoint of each highway (and from t as well) until they hit a road as in Figure 2. Thus we obtain a horizontal trapezoidal map $T(\mathcal{H}, t)$ that decomposes the plane into rectangles. Let $\mathcal{A}(t, \mathcal{H})$ be the subdivision of the plane induced by the arrangement of the set of segments $T(\mathcal{H}, t) \bigcup T^\ell(\mathcal{H}, t)$ (such as the one in Figure 3).

We can observe the following:

Lemma 4 *For any cell c, the value $k_\beta = |E_t(\beta)|$ is independent of choice of $\beta \in c$. Thus, we write k_c for k_β. For any two points p, q in the same cell $c \in \mathcal{A}(t, \mathcal{H})$, their corresponding highways h_p and h_q intersect exactly the same horizontal edges in $T(\mathcal{H}, t)$. Thus, the y coordinate of β_i is independent of choice of $\beta \in c$ for $i = 2, 3, \ldots, k_c - 1$.*

Lemma 5 *The function $g_{i,j}(\beta) = g_{i,j}(\xi, \eta)$ is piecewise linear within each cell c if $i, j \leq k_c$.*

Proof. Recall that the x-value of β_i equals ξ. From Lemma 4, within a cell c, the y-value of β_i is constant if $i \neq 1, k_c$, and linear in η if $i = 1$ or $i = k_c$. Since the $d_{\mathcal{H}}$ shortest path distance from a fixed point (say, s) to a point is linear within a cell of the shortest path map, all terms $d_{\mathcal{H}}(s, \beta_i) + d(\beta_i, \beta_j)/v + d_{\mathcal{H}}(\beta_j, t)$ are piecewise linear, and hence $g_{i,j}(\beta)$ is piecewise linear if $\beta \in c$. □

Given $s \in \mathcal{S}$, $t \in \mathcal{T}$, and $\mathcal{H}$, we say that a subdivision $\mathcal{R}$ is an admissible s-t-subdivision if it satisfies the following:

1. $\mathcal{R}$ is a refinement of $\mathcal{A}(t, \mathcal{H})$
2. Fixed any cell $C \in \mathcal{R}$ such that $C \subseteq c$ for a cell c of $\mathcal{A}(t, \mathcal{H})$. Then, for every $\beta, \beta' \in C$, and $i \in [1, k_c]$, points β_i and β'_i are in the same cells of $SPM(s, \mathcal{H})$ and $SPM(t, \mathcal{H})$.

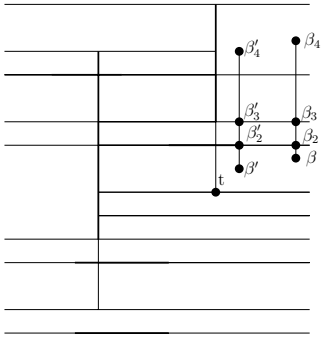


Figure 3: The y coordinates of exit points do not depend on the location of β in a cell of $\mathcal{A}(t, \mathcal{H})$

Lemma 6 *For any cell c of $\mathcal{A}(t, \mathcal{H})$ and $i, j \in [1, k_c]$, the function $g_{i,j}(x, y)$ is linear within each refined cell $C \subset c$ of an admissible s-t-subdivision $\mathcal{R}$.*

Once we have an admissible s-t-subdivision we can compute $f_{s,t}(x, y)$ as follows:

Lemma 7 *Suppose that we have an admissible s-t-subdivision of complexity k. Then each cell is further refined into $O(1)$ cells such that $f^{new}_{s,t}(x, y)$ is linear in each subcell. This refinement can be computed in $O(k)$ time.*

Thus, our task is to obtain an admissible s-t-subdivision with a low complexity. The first and third terms in the expression $d_{\mathcal{H}}(s, \beta_i) + d(\beta_i, \beta_j)/v + d_{\mathcal{H}}(\beta_j, t)$ are controlled by $SPM(s, \mathcal{H})$ and $SPM(t, \mathcal{H})$ within a cell c of $\mathcal{A}(t, \mathcal{H})$, respectively. Thus, we overlay the planar subdivisions $SPM(s, \mathcal{H})$, $SPM(t, \mathcal{H})$ and $\mathcal{A}(t, \mathcal{H})$, refine if necessary, to obtain an admissible s-t-subdivision.

4.3 Computing an admissible s-t-subdivision

Given $s \in \mathcal{S}$, $t \in \mathcal{T}$, let $I(s, t)$ be the arrangement induced by the overlay of regions $SPM(s, \mathcal{H})$, $SPM^{\ell}(s, \mathcal{H})$, $SPM(t, \mathcal{H})$, $SPM^{\ell}(t, \mathcal{H})$, $T(\mathcal{H}, t)$ and $T^{\ell}(\mathcal{H}, t)$, that is: two points p and q are in the same cell of $I(s, t)$ if and only if they are in the same cell of each of these subdivisions.

Lemma 8 *Given $s \in \mathcal{S}$, $t \in \mathcal{T}$, and $\mathcal{H}$, the subdivision $I(s, t)$ has complexity $O(H^2)$ (and can be computed in $O(H^2)$ time).*

Obviously, $I(s, t)$ is a refinement of $\mathcal{A}(t, \mathcal{H})$. If h is a railway, the only possible entrance and exit points are the endpoints of h, and we can observe that the second condition of an admissible s-t-subdivision also holds. Namely, we have the following:

Theorem 9 *If h is a railway, $I(s, t)$ is an admissible s-t-subdivision.*

For the case where h is a road, we need to further refine $I(s, t)$. From Lemma 4, β_i is on a horizontal segment at a fixed height $\nu_{c,i}$ if $\beta \in c$ and $1 < i < k_c$. We trace the changes of $SPM(s, \mathcal{H})$ and $SPM(t, \mathcal{H})$ at each height $\nu_{c,i}$: Consider the piecewise linear function $f_t(x, y) = d_{\mathcal{H}}((x, y), t)$ that is the distance between t and $q = (x, y)$. Consider the one dimensional function $f_t(x, \nu_{c,i})$, piecewise linear if $(x, \nu_{c,i}) \in c$. Let $\mathcal{B}_t(c, i)$ be the set of breakpoints of $f_t(x, \nu_{c,i})$ for a fixed c and $1 < i < k_c$. We cut each cell $c \in I(s, t)$ into subcells by all vertical lines that pass through points in $\bigcup_{1<i<k_c} \mathcal{B}_t(c, i)$. Let $I^*(s, t)$ be the resulting subdivision. We can prove the following lemma:

Theorem 10 *If h is a road, $I^*(s, t)$ is an admissible s-t-subdivision of complexity $O(H^3)$ that can be computed in $O(H^3)$ time.*

This leads us to the main result of this paper:

Theorem 11 *Given $\mathcal{S}$, $\mathcal{T}$ and $\mathcal{H}$, the optimal location of a new isothetic segment highway of length ℓ for any of the nine objective functions can be calculated in polynomial time. The time complexities in Table 1 follows from Theorems 9, 10 and Table 2.*

References

[1] H.-K. Ahn, H. Alt, T. Asano, S. W. Bae, P. Brass, O. Cheong, C. Knauer, H.-S. Na, C.-S. Shin, A. Wolff, *Constructing Optimal Highways*. Computing: The Australian Theory Symposium (CATS'07) (2007), Vol. 65.

[2] S. W. Bae, J.-H. Kim, *Optimal Construction of the City Voronoi Diagram*. Proc. 17th Annu. Internat. Sympos. Algorithms Comput. (2006), pp. 183-192.

[3] O. Aicholzer, F. Aurenhammer and B. Palop, *Quickest path, straight skeleton and the city Voronoi diagram*. ACM Sympos. Comput. Geom (2002). pp. 151-159.

[4] J.S.B. Mitchell, *L_1 shortest paths among obstacles in the plane*. Internat. J. Comput. Geom. Appl. 6(3). pp. 309-331 (1996).

[5] M. Abellanas, F. Hurtado, C. Icking, R. Klein, E. Langetepe, L. Ma, B. Palop del Río, V. Sacristan, *Proximity problems for time metrics induced by the L_1 metric and isothetic networks*. Encuentros en Geometría Computacional, N. Coll et al. (eds.), Universitat de Girona (2001), pp. 175-181.

[6] J. Cardinal, S. Langerman, *Min-max-min Geometric facility location Problems*. Proc. of the European Workshop on Computational Geometry (2006), Delphi, pp. 149-152.

[7] J. Cardinal, S. Collette, F. Hurtado, S. Langerman, and B. Palop, *Moving Walkways, Escalators, and Elevators*. Eprint arXiv:0705.0635 (2007).

[8] M. Sharir, P.K. Agarwal, *Davenport-Schinzel Sequences and Their Geometric Applications*. Cambridge University Press (1995), pp. 175-245.

Algorithms for Graphs of Bounded Treewidth via Orthogonal Range Searching

Sergio Cabello* Christian Knauer†

Abstract

We show that, for any fixed constant $k \geq 3$, the sum of the distances between all pairs of vertices of an abstract graph with n vertices and treewidth at most k can be computed in $O(n \log^{k-1} n)$ time.

We also show that, for any fixed constant $k \geq 2$, the dilation of a geometric graph (i.e., a graph drawn in the plane with straight-line segments) with n vertices and treewidth at most k can be computed in $O(n \log^{k+1} n)$ expected time.

The algorithms for both problems are based on the same principle: data structures for orthogonal range searching in bounded dimension provide a compact representation of distances in abstract graphs of bounded treewidth.

1 Introduction

Let $G = (V, E)$ be a graph with n vertices and assume that each edge of E has an associated nonnegative abstract length $\ell(e)$. We can define the length of a path in G as the sum of the lengths of its edges. The shortest path distance $d_G(u, v)$ between any pair of vertices u, v is defined as the minimum length over all walks in G between u, v. We are interested in the sum over all ordered pairs of vertices of their distance, that is,

$$\Sigma(G) = \sum_{(u,v)\in V^2} d_G(u, v).$$

If the length of each edge is one, the value $\frac{1}{2}\Sigma(G)$ is known as the *Wiener index* of G, which is a generalization of the original definition given by Wiener in 1947 [21]. *Molecular topological indices* are values defined by the graph-model of a molecule with the hope that they correlate with physical and chemical properties of the molecules [20]. The Wiener index is probably the most studied molecular topological index, with over thousand publications.

From the algorithmical point of view, the main question is what classes of graphs do not require to compute all the pairwise distances to obtain the Wiener index, or more generally, the value $\Sigma(G)$. Linear time algorithms are known for trees [16], cacti [23], and benzenoid systems[1] [10, 11]. One of the main algorithmical open problems in this context concerns the existence of subquadratic algorithms for computing the Wiener index of planar graphs.

The average distance of a graph and $\Sigma(G)$ are essentially the same object, and have been studied in other models. For abstract discrete metric spaces, given by a matrix of distances, Indyk [14] gives a sublinear $(1 + \varepsilon)$-approximation algorithm based on sampling. Note that this model is substantially different, since it assumes that any distance is available at constant time, which does not hold in general graphs. The well-separated pair decomposition [7] can be used to obtain deterministic $(1+\varepsilon)$-approximations to the average distance in Euclidean spaces or, more generally, in spaces of bounded doubling dimension [13].

A *geometric graph* is a graph whose vertex set is a finite set of points, and where the weight/length of each edge equals the Euclidean distance between its vertices. The *dilation* (or stretch-factor) $\Delta(G)$ of a geometric graph G is the largest ratio between the distance d_G and the Euclidean distance:

$$\Delta(G) := \max_{u,v\in V(G),u\neq v} \frac{d_G(u, v)}{||u - v||}.$$

Substantial research has been done about constructing so-called geometric spanners: geometric graphs with small dilation, few edges, and other additional properties; see the monograph [18]. Here, we turn our attention to a different problem: computing the dilation of a given geometric graph.

One can trivially compute the distance between all pairs of vertices, and then compute the dilation. However, this approach neglects all the geometry of the problem, and the question of whether one actually needs to compute all distances naturally arises. Agarwal *et al.* [1] give near-linear time algorithms for computing the exact dilation of geometric paths, cycles, and trees. For computing a $(1 + \varepsilon)$-approximation to the dilation, assuming that $\varepsilon > 0$ is constant, Narasimhan and Smid [17] show that the problem reduces in $O(n \log n)$ time to compute the graph dis-

*Department of Mathematics, IMFM, and Department of Mathematics, FMF, University of Ljubljana, Slovenia. E-mail: sergio.cabello@fmf.uni-lj.si. Research supported by the Slovenian Research Agency, project J1-7218.

†Institut für Informatik, Freie Universität Berlin, Takustraße 9, D–14195 Berlin, Germany. E-mail: christian.knauer@inf.fu-berlin.de.

[1] Benzenoid systems are subgraphs of the regular hexagonal grid enclosed by a circuit.

tance between $O(n)$ pairs of vertices. Combining this result with the data structure of Chaudhuri and Zaroliagis [9], one obtains, for any fixed $\varepsilon > 0$, a $(1+\varepsilon)$-approximation algorithm for the dilation that runs in $O(n \log n)$ time.

Our results. We show that the sum of distances of an abstract graph and the dilation of a geometric graph can be computed in near-linear time when the graphs have bounded treewidth. More precisely:

- for any fixed constant $k \geq 3$, the sum of distances $\Sigma(G)$ can be computed in $O(n \log^{k-1} n)$ time for any abstract graph with n vertices and treewidth at most k. See Section 4.
- for any fixed constant $k \geq 2$, the dilation $\Delta(G)$ can be computed in $O(n \log^{k+1} n)$ expected time for any geometric graph with n vertices and treewidth at most k. Due to space constraints we will omit the proof of this result from this abstract; details can be found in [6].

Similar results were only known for graphs of treewidth 1 (trees) and some subclasses of graphs with treewidth 2 (cycles and cacti), c.f. [1, 23]. Our algorithms are based on divide-and-conquer, using the fact that graphs with bounded treewidth have balanced small separators. Although most algorithms that use treewidth are based on dynamic programming, this approach does not seem appropriate here.

The approach in both cases is the same: distances in abstract graphs of treewidth k are closely related to orthogonal range searching in $\mathbb{R}^{k-1}$; see Section 3. This approach is implicit in [2] and more explicitly mentioned by Shi [19]. This relation allows to use techniques from computational geometry to design efficient algorithms for problems involving distances in graphs of bounded treewidth.

2 Toolbox

Treewidth. Treewidth is a parameter measuring, in some sense, the complexity of a graph. We next give its definition; see Bodlaender [5] for an overview.

Definition 1 *A* tree decomposition *of a graph G is a pair (X, T), where $X = \{X_i \subseteq V(G) \mid i \in I\}$ is a collection of subsets of $V(G)$ (called* bags*), and a tree $T = (I, F)$ with a node set I such that:* (i) $V(G) = \bigcup_{i \in I} X_i$*;* (ii) *for every edge $uv \in E(G)$ there is some bag $X_i \in X$ such that $u, v \in X_i$;* (iii) *for all $u \in V(G)$, the nodes $\{i \in I \mid u \in X_i\}$ form a connected subtree of T. The* width *of a tree decomposition $(\{X_i \mid i \in I\}, T)$ is $\max_{i \in I} |X_i| - 1$. The* treewidth *of G is the minimum width over all tree decompositions of G.*

It is known that graphs of treewidth k have $O(kn)$ edges, an thus graphs of bounded treewidth have $O(n)$ edges. Computing the treewidth of a graph is NP-hard. However, for any fixed constant $k > 0$, we can decide in linear time if a given graph has treewidth at most k, and in such case, construct a tree decomposition of linear size and width k in linear time [4]. Our main results apply assuming that we have graphs of bounded treewidth.

Our approach will be based on divide-and-conquer. We will use the following concept, closely related to separators.

Definition 2 *Let A be a subset of vertices of the graph G. The* portals *of A are the vertices of A that have some edge incident to $V(G) \setminus A$.*

A standard result for graphs with treewidth at most k is the existence of $(2/3)$-separators of size $k+1$. In our approach, the number of vertices in the separator is very relevant, and therefore we will reduce the size of the separator by making the separation more unbalanced. A sketch of the following result can be found in the notes of Biedl [3].

Lemma 1 *Let $k \geq 1$ be a constant. Given a graph G with $n > k+1$ vertices and treewidth at most k, we can find in linear time a subset of vertices $A \subseteq V(G)$ such that:*

(i) *A has between $\frac{n}{k+1}$ and $\frac{nk}{k+1}$ vertices;*

(ii) *A has at most k portals;*

(iii) *adding edges between the portals of A does not change the treewidth.*

Assume that we have a graph G with treewidth k and a subset of vertices A with the properties stated in Lemma 1. Let S be the set of portals of A and let $B = (V(G) \setminus A) \cup S$. Note that the set of portals of B is a subset of S.

Consider the graph G' obtained from G by adding an edge ss' with weight $d_G(s, s')$ between each pair $s, s' \in S$. If G' has multiple edges, we only keep the edges with minimum weight. Finally, let G_A denote the subgraph of G' induced by A. From the property (iii) of A, we know that G' has treewidth k, and thus G_A has treewidth at most k. Furthermore, it is straightforward to see that $d_G(a, a') = d_{G_A}(a, a')$ for any $a, a' \in A$. Using the notation $B = (V(G) \setminus A) \cup S$, the same argument applies to G_B: it has treewidth at most k and $d_G(b, b') = d_{G_B}(b, b')$ for all $b, b' \in B$.

Orthogonal range searching. Let P be a set of points in $\mathbb{R}^d$. Assume we are given a function $w : P \to \mathbb{R}$ that assigns a weight $w(p)$ to each point $p \in P$. We extend the weight function to any subset Q of points by $w(Q) := \sum_{p \in Q} w(p)$. A *rectangle* R in $\mathbb{R}^d$ is the Cartesian product of d intervals, $R = I_1 \times \cdots \times I_d$,

where each interval I_i can include both extremes, one of them, or none.

Orthogonal range searching deals with the problem of preprocessing P such that, for a query rectangle R, certain properties of $P \cap R$ can be efficiently reported. We will use the following two results.

Theorem 2 ([22]) *Let $d \geq 2$ be a constant. Given a set of n points $P \subset \mathbb{R}^d$ and a weight function $w : P \to \mathbb{R}$, there is a data structure that can be constructed in $O(n \log^{d-1} n)$ time such that, for any query rectangle R, the weight $w(P \cap R)$ can be reported in $O(\log^{d-1} n)$ time.*

3 Distances and orthogonal range searching

Let G be a graph and let A be a subset of its vertices with k portals S, enumerated as $s_1, \ldots, s_k$. Let $B = (V(G) \setminus A) \cup S$. We use the notation $[k] = \{1, \ldots, k\}$. For any vertex $b \in B$ and any index $i \in [k]$, let $A(b, i)$ be the subset of vertices $a \in A$ such that:

(i) There exists a shortest path from b to a through s_i, that is, $d_G(a, b) = d_G(a, s_i) + d_G(s_i, b)$.

(ii) There is no shortest path from b to a through s_j for $j < i$, that is, $d_G(a, b) < d_G(a, s_j) + d_G(s_j, b)$ for all $j \in [i-1]$.

For any $b \in B$, the union of $A(b, 1), \ldots, A(b, k)$ is the whole A. We include (ii) to ensure that the sets $A(b, 1), \ldots, A(b, k)$ are pairwise disjoint, which will be relevant for not over counting in Section 4. Note that different enumerations of the portals may give completely different sets $A(b, 1), \ldots, A(b, k)$, not just a reordering.

Assume that we have a weight function $\phi : A \to \mathbb{R}$ assigning a weight to each vertex of A. We extend the weight function to any subset $A' \subseteq A$ by $\phi(A') := \sum_{a \in A'} \phi(a)$. For each $i \in [k]$, we can use orthogonal range searching to preprocess the graph G so that, for any query vertex $b \in B$, information concerning $A(b, i)$ can be reported quickly and in a compact form.

Theorem 3 *Let $k \geq 3$ be a constant. Assume we are given a graph G with n vertices and m edges, a subset of vertices A with k portals S enumerated $s_1, \ldots, s_k$, a weight function $\phi : A \to \mathbb{R}$, and let $B = (V(G) \setminus A) \cup S$. For any given $i \in [k]$, there is a data structure that can be constructed in $O(m + n \log^{k-2} n)$ time such that, for any query vertex $b \in B$, the weight $\phi(A(b, i))$ can be reported in $O(\log^{k-2} n)$ time.*

Proof. We first construct a shortest path tree from each of the portals $s_1, \ldots, s_k$ and store the values $d_G(s_j, v)$ for all $j \in [k]$ and $v \in V(G)$. Since we assume $k = O(1)$, we spend $O(m + n \log n)$ time for computing these shortest path trees.

For each vertex $a \in A$ we define a point $p_a \in \mathbb{R}^k$ with coordinates $p_a(j) = d_G(a, s_i) - d_G(a, s_j)$, where $p_a(j)$ denotes the j-th coordinate of point p_a. Let P be the set of points $p_a, a \in A$. Note that the i-th coordinate of the points in P is always 0, and therefore we can regard P as a set of $|A|$ points in $\mathbb{R}^{k-1}$. We define a weight function for each point $p_a \in P$ by $w(p_a) := \phi(a)$. Clearly, we have $\phi(A') = w(\{p_a \in P \mid a \in A'\})$. Finally, we preprocess the point set P with weight w into a data structure as described in Theorem 2, where $d = k - 1$. This finishes the description of the data structure. Preprocessing P takes $O(|A| \log^{k-2} |A|) = O(n \log^{k-2} n)$ time, and hence we spend $O(n \log^{k-2} n)$ time to construct the data structure.

When we receive a query $b \in B$, we proceed as follows. For $j \in [k]$ define the interval $I_j(b)$ by

$$I_j(b) = \begin{cases} (-\infty,\ d_G(s_j, b) - d_G(s_i, b)) & \text{if } j < i, \\ (-\infty,\ +\infty) & \text{if } j = i, \\ (-\infty,\ d_G(s_j, b) - d_G(s_i, b)] & \text{if } j > i. \end{cases}$$

Consider the rectangle $R(b) = I_1(b) \times \cdots \times I_k(b)$.

Any path from $a \in A$ to b has to use some portal of A, and hence the condition $d_G(a, b) = d_G(a, s_i) + d_G(s_i, b)$ can be rewritten as

$$d_G(a, s_i) + d_G(s_i, b) \leq d_G(a, s_j) + d_G(s_j, b) \quad \forall j \in [k].$$

It is now easy to verify that

$$A(b, i) = \{a \in A \mid p_a \in R(b)\}.$$

Since

$$\phi(A(b, i)) = \phi(\{a \in A \mid p_a \in R(b)\}) = w(P \cap R(b)),$$

we can obtain $\phi(A(b, i))$ in $O(\log^{k-2} n)$ time by querying the data structure storing P for the value $w(P \cap R(b))$. □

4 Sum of distances

We are interested in computing $\Sigma(G)$ for a weighted graph G whose treewidth is bounded by a constant $k \geq 3$. Let A be a set of vertices of G obtained by Lemma 1. Like before, we enumerate the (at most) k portals S of A by $s_1, \ldots, s_k$, and set $B = (V(G) \setminus A) \cup S$. Recall the definition of G_A and G_B, given after Lemma 1. Using the notation $\Sigma(C, C') = \sum_{c \in C} \sum_{c' \in C'} d_G(c, c')$ for any $C, C' \subseteq V(G)$, we have

$$\begin{aligned} \Sigma(G) =& \Sigma(G_A) + \Sigma(G_B) + \\ & 2 \cdot (\Sigma(A, B) - \Sigma(S, A) - \Sigma(S, B)). \end{aligned} \quad (1)$$

Lemma 4 *We can compute $\Sigma(A, B)$ in $O(n \log^{k-2} n)$ time.*

Proof. For any $b \in B$, we know that A is the disjoint union of $A(b,1), A(b,2), \ldots, A(b,k)$. Define weight functions $\phi_0, \phi_1, \ldots \phi_k : A \to \mathbb{R}$ by $\phi_0(a) = 1$ for all $a \in A$ and by $\phi_i(a) = d_G(s_i, a)$ for $a \in A, i \in [k]$. We can then write $\Sigma(A,B)$ as

$$\sum_{b \in B} \sum_{i=1}^{k} \left[\phi_0(A(b,i)) \cdot d_G(s_i,b) + \phi_i(A(b,i))\right]. \qquad (2)$$

We now use Theorem 3 several times: for each $i \in [k]$ we make a data structure $DS_0^{(i)}$ for weight ϕ_0 and a data structure $DS_1^{(i)}$ for weight ϕ_i. We construct $2k = O(1)$ data structures, and each one takes $O(n \log^{k-2} n)$ preprocessing time. Any value $\phi_0(A(b,i))$ or $\phi_i(A(b,i))$ can now be obtained in time $O(\log^{k-2} n)$ by querying the appropriate data structure. Therefore, we can get the values $\phi_0(A(b,i))$ and $\phi_i(A(b,i))$ for all $(b,i) \in B \times [k]$ in time $O(k|B| \log^{k-2} n) = O(n \log^{k-2} n)$. Finally, the values $d_G(s_i,b)$ for all $(s_i,b) \in S \times B$ can be obtained in $O(n \log n)$ time constructing a shortest path tree from each portal $s_i \in S$, and thus we can compute $\Sigma(A,B)$ using (2). □

Theorem 5 *Let $k \geq 3$ be a constant. Given a graph G with n vertices and treewidth at most k, we can compute $\Sigma(G)$ in $O(n \log^{k-1} n)$ time.*

Proof. If G has $O(1)$ vertices, we compute $\Sigma(G)$ by brute force. Otherwise we find a set A as described in Lemma 1. The values $\Sigma(S,A)$ and $\Sigma(S,B)$ are computed using shortest path trees from each portal $s \in S$, while $\Sigma(A,B)$ is computed using Lemma 4. The graphs G_A and G_B can be constructed using shortest path trees from each $s \in S$, and we can then recursively compute $\Sigma(G_A)$ and $\Sigma(G_B)$. Finally, we use the relation (1) to compute the value $\Sigma(G)$. The conquer step can be done in $O(n \log^{k-2} n)$ time. Since we assume that k is constant, the recursion is balanced because of Lemma 1(i), and the result follows. □

References

[1] P. K. Agarwal, R. Klein, C. Knauer, S. Langerman, P. Morin, M. Sharir, and M. Soss. Computing the maximum detour and spanning ratio of 2- and 3-dimensional paths, trees, and cycles. *Discrete Comput. Geom.*, to appear.

[2] B. Ben-Moshe, B. K. Bhattacharya, and Q. Shi. Efficient algorithms for the weighted 2-center problem in a cactus graph. In *ISAAC 2005*, volume 3827 of *LNCS*, pages 693–703, 2005.

[3] T. Biedl. Lecture 12: Separators in partial k-trees (feb. 13, 2004). Lecture notes for the course CS 762 (Graph-Theoretic Algorithms), Winter 2004, University of Waterloo.

[4] H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.

[5] H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.*, 209:1–45, 1998.

[6] S. Cabello and C. Knauer. Algorithms for graphs of bounded treewidth via orthogonal range searching. *Technical report B-08-01*, Dep. of Computer Science, Freie Universität Berlin, 2008.

[7] P. B. Callahan and S. R. Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *J. ACM*, 42(1):67–90, 1995.

[8] T. M. Chan. Geometric applications of a randomized optimization technique. *Discrete Comput. Geom.*, 22(4):547–567, 1999.

[9] S. Chaudhuri and C. D. Zaroliagis. Shortest paths in digraphs of small treewidth. Part I: Sequential algorithms. *Algorithmica*, 27:212–226, 2000.

[10] V. Chepoi and S. Klavžar. The Wiener index and the Szeged index of benzenoid systems in linear time. *J. Chem. Inf. Comput. Sci.*, 37:752–755, 1997.

[11] V. Chepoi and S. Klavžar. Distances in benzenoid systems: further developments. *Discrete math.*, 192:27–39, 1998.

[12] S. J. Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2:153–174, 1987.

[13] S. Har-Peled and M. Mendel. Fast construction of nets in low dimensional metrics, and their applications. *SIAM J. Comput.*, 35(5):1148–1184, 2006.

[14] P. Indyk. Sublinear time algorithms for metric space problems. In *STOC '94*, pages 428–434, 1999.

[15] D. G. Kirkpatrick. Optimal search in planar subdivisions. *SIAM J. Comput.*, 12(1):28–35, 1983.

[16] B. Mohar and T. Pisanski. How to compute the Wiener index of graph. *J. Math. Chem.*, pages 267–277, 1988.

[17] G. Narasimhan and M. Smid. Approximating the stretch factor of Euclidean graphs. *SIAM J. Comput.*, 30:978–989, 2000.

[18] G. Narasimhan and M. Smid. *Geometric Spanner Networks*. Cambridge University Press, 2007.

[19] Q. Shi. Single facility location problems in partial k-trees, 2005. Poster at MITACS, Canada.

[20] N. Trinajstić. *Chemical Graph Theory*. CRC Press, 2nd edition, 1992.

[21] H. Wiener. Structural determination of paraffin boiling points. *J. Amer. Chem. Sot.*, 69:17–20, 1947.

[22] D. E. Willard. New data structures for orthogonal range queries. *SIAM J. Comput.*, 14:232–253, 1985.

[23] B. Zmazek and J. Žerovnik. Computing the weighted Wiener and Szeged number on weighted cactus graphs in linear time. *Croatica Chemica Acta*, 76:137–143, 2003.

A Tight Bound for the Delaunay Triangulation of Points on a Polyhedron

Nina Amenta* Dominique Attali† Olivier Devillers‡

Abstract

We show that the Delaunay triangulation of a set of n points distributed nearly uniformly on a p-dimensional polyhedron (not necessarily convex) in d-dimensional Euclidean space is $O(n^{\frac{d-k+1}{p}})$, where $k = \lceil \frac{d+1}{p+1} \rceil$. This bound is tight, and improves on the prior upper bound for most values of p.

1 Introduction

Overview. The Delaunay triangulation of a set of points is a fundamental geometric data structure, used, in low dimensions, in surface reconstruction, mesh generation, molecular modeling, geographic information systems, and many other areas of science and engineering. In higher dimensions, it is well-known [9] that the complexity of the Delaunay triangulation of n points is $O(n^{\lceil \frac{d}{2} \rceil})$ and that this bound is achieved by distributions of points along one-dimensional curves such as the moment curve. But points distributed uniformly in $\mathbb{R}^d$, for instance inside a d-dimensional ball, have Delaunay triangulations of complexity $O(n)$; the constant factor is exponential in the dimension, but the dependence on the number of points is linear. In an earlier paper [1], we began to fill in the picture in between these two extremes, that is, when the points are distributed on a manifold of dimension $2 \leq p \leq d-1$. We began with the easy case of a *p-dimensional polyhedron* P, and showed that for a particular (probably overly restrictive) sampling model the size of the Delaunay triangulation is $O(n^{(d-1)/p})$.

Main result. Here as in [1], we consider a fixed *p-dimensional polyhedron* P in d-dimensional Euclidean space $\mathbb{R}^d$. Our point set S is a *sparse ε-sample* from P. Sparse ε-sampling requires the sampling to be neither too sparse nor too dense. Let n be the number of points in S. We consider the complexity of the Delaunay triangulation of S, as $n \to \infty$, while P remains fixed. The main result in this paper is that the number of simplices of all dimensions is $O(n^{\frac{d-k+1}{p}})$ where $k = \lceil \frac{d+1}{p+1} \rceil$. The hidden constant factor depends, among other things, on the geometry of P, which is constant since P is fixed.

At the coarsest level, the idea of this proof is the same as that of [1]: we map Delaunay simplices to the medial axis and then use a packing argument to count them. The key new idea is the observation that when $k = \lceil \frac{d+1}{p+1} \rceil > 2$, the vertices of any Delaunay simplex, which must span $\mathbb{R}^d$, have to be drawn from more than two faces of P. This allows us to map Delaunay simplices to only the lower-dimensional submanifolds of the medial axis, induced by k or more faces. To realize this scheme, we introduce a new geometric structure, the *quasi medial axis*, which replaces the centers of tangent balls defining the medial axis with the centers of tangent annuli. In this paper, we only present an outline of the proof. Full details can be found in [2].

Prior work. The complexity of the Delaunay triangulation of a set of points on a two-manifold in $\mathbb{R}^3$ has received considerable recent attention, since such point sets arise in practice, and their Delaunay triangulations are found nearly always to have linear size. Golin and Na [6] proved that the Delaunay triangulation of a large enough set of points distributed uniformly at random on the surface of a fixed convex polytope in $\mathbb{R}^3$ has expected size $O(n)$. They later [7] established an $O(n \log^6 n)$ upper bound with high probability for the case in which the points are distributed uniformly at random on the surface of a non-convex polyhedron.

Attali and Boissonnat considered the problem using a sparse ε-sampling model similar to the one we use here, rather than a random distribution. For such a set of points distributed on a polygonal surface P, they showed that the size of the Delaunay triangulation is $O(n)$ [3]. In a subsequent paper with Lieutier [4] they considered "generic" smooth surfaces, and got an upper bound of $O(n \log n)$. Specifically, a "generic" surface is one for which each medial ball touches the surface in at most a constant number of points.

The genericity assumption is important. Erickson considered more general point distributions, which he characterized by the *spread*: the ratio of the largest inter-point distance to the smallest. The spread of a sparse ε-sample of n points from a two-dimensional manifold is $O(\sqrt{n})$. Erickson proved that the De-

*Computer Science Department, University of California, amenta@ucdavis.edu

†Gipsa-lab, CNRS Grenoble, Dominique.Attali@gipsa-lab.inpg.fr

‡INRIA, Sophia-Antipolis , Olivier.Devillers@sophia.inria.fr

launay triangulation of a set of points in $\mathbb{R}^3$ with spread Δ is $O(\Delta^3)$. Perhaps even more interestingly, he showed that this bound is tight for $\Delta = \sqrt{n}$, by giving an example of a sparse ε-sample of points from a cylinder that has a Delaunay triangulation of size $\Omega(n^{3/2})$ [5]. Note that this surface is not generic and has a degenerate medial axis.

To the best of our knowledge, ours [1] is the only prior result for $d > 3$.

2 Statement of Theorem

In this section, we introduce the setting for our result. Given a polyhedron $P \subseteq \mathbb{R}^d$ and a point x on P, let F_x be the unique face that contains x. We say that a set of points $S \subseteq P$ is a *λ-sparse ε-sample* of P iff it satisfies the following two conditions:

`Density:` Every point x in P is at distance ε or less to a point s in S lying on the closure of F_x.

`Sparsity:` Every closed d-ball with radius $6d\varepsilon$ contains at most λ points of S.

Note that our density condition implies that all faces of all dimensions are uniformely sampled, not just faces with highest dimension as in [3, 4]. Hereafter, we consider λ to be a constant. The number n of points in a λ-sparse ε-sample of a p-dimensional polyhedron is related to ε by $n = \Theta(\varepsilon^{-p})$. Thus, as n tends to infinity, ε tends to zero. We are now ready to state our main result:

Theorem 1 *Let S be a λ-sparse ε-sample of a p-dimensional polyhedron P in $\mathbb{R}^d$, and let n be the number of points in S. The Delaunay triangulation of S has size $O(n^{\frac{d-k+1}{p}})$ where $k = \lceil \frac{d+1}{p+1} \rceil$.*

Note that our result requires no non-degeneracy assumption, neither on P nor on S.

3 Essential quasi medial axes

In this section, we introduce the *ε-quasi k-medial axis*, a variant of the medial axis based on tangent annuli rather than tangent balls, which is the key geometric object in our proof. We then define the part of the ε-quasi k-medial axis to which Delaunay simplices will be mapped: the essential ε-quasi k-medial axis (considering only the parts of dimension at most $d - k + 1$ and lopping off the parts which extends to infinity). Along the way, we give a tool to identify lower-dimensional parts of the ε-quasi k-medial axis.

3.1 Quasi medial axes

We start by defining ε-quasi k-medial axes. We say that a $(d-1)$-sphere Σ is *tangent* to a face F at point x if both the closure of F and the affine space spanned by F intersect Σ in a unique point x. An annulus with center z, inner radius r and outer radius R is the set of points x whose distance to the center satisfies $r \leq \|x-z\| \leq R$. The boundary of an annulus consists of two $(d-1)$-spheres and we call the smallest one the *inner sphere* and the largest one the *outer sphere*. Extending what we just defined for spheres, we say that an annulus A is *tangent* to F at x if one of the two spheres bounding A is tangent to F at x. Point x is called a *tangency point* of A. An annulus is *P-empty* if its inner sphere bounds a d-ball whose interior does not intersect P. An annulus is called *ε-thin* if the difference between the outer and inner radii squared is $R^2 - r^2 = \varepsilon^2$.

Definition 1 *The ε-quasi k-medial axis $\mathcal{M}^k(P, \varepsilon)$ of P is the set of points $z \in \mathbb{R}^d$ for which for the largest P-empty ε-thin annulus centered at z, $A(z, \varepsilon)$, is tangent to at least k faces of P (see Figure 1).*

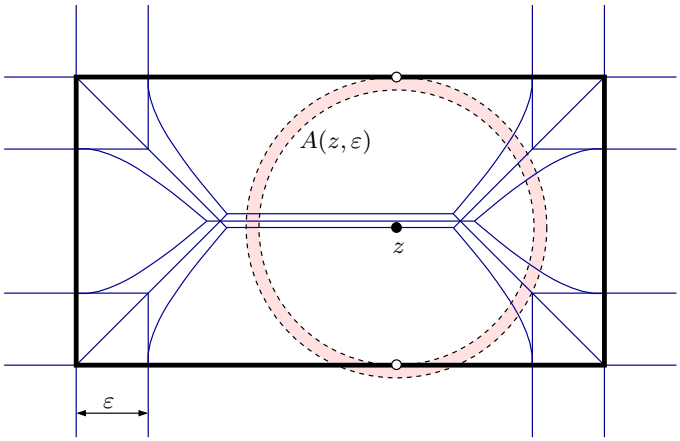


Figure 1: A rectangle and its ε-quasi 2-medial axis composed of 16 half-lines, 5 segments and 8 pieces of hyperbolas.

3.2 Identifying lower-dimensional strata

Because P might be degenerate, we must introduce a tool to identify the parts of $\mathcal{M}^k(P, \varepsilon)$ which have dimension $d - k + 1$ or less very carefully. We recall that a stratification of a subset $X \subseteq \mathbb{R}^d$ is a filtration

$$\emptyset = X_{-1} \subseteq X_0 \subseteq \cdots \subseteq X_j = X$$

by subspaces such that the set difference $X_i \setminus X_{i-1}$ is a i-dimensional manifold, called the i-dimensional *stratum* of X. In particular, semi-algebraic sets admit a stratification [8] and since ε-quasi k-medial axes of polyhedra are piecewise semi-algebraic, they also admit a stratification.

Definition 2 *We say that k faces $F_1, \ldots, F_k$ are* independent *if none of them is contained in the affine*

space spanned by the union of the others, that is for $1 \leq i \leq k$,

$$F_i \not\subseteq \text{Aff}(F_1 \cup \cdots \cup \widehat{F_i} \cup \cdots \cup F_k),$$

where the symbol $\widehat{\ }$ *over* F_i *indicates that it is omitted in the union.*

Lemma 2 *Let* $z \in \mathcal{M}^k(P,\varepsilon)$ *and suppose that* $A(z,\varepsilon)$ *is tangent to* j *faces amongst which* k *faces are independent. Then,* z *lies on a stratum of* $\mathcal{M}^k(P,\varepsilon)$ *of dimension* $d-k+1$ *or less.*

3.3 Essential part

The ε-quasi k-medial axis in general extends to infinity and therefore can have an infinite volume. In this section, we select a subset of the ε-quasi k-medial axis called the essential ε-quasi k-medial axis, $\bar{\mathcal{M}}^k(P,\varepsilon)$ in such a way that all its strata will have a finite volume bounded by a constant that does not depend on ε. For this, we need some definitions. We say that a hyperplane *supports* $X \subseteq \mathbb{R}^d$ if it has non-empty intersection with the boundary of X and empty intersection with the interior of X.

Definition 3 *A point* z *is* ε-essential *if there exists no hyperplane supporting the convex hull of* P *and containing all faces tangent to* $A(z,\varepsilon)$.

It follows immediately from the definition that:

Lemma 3 *If the union of faces tangent to* $A(z,\varepsilon)$ *spans* $\mathbb{R}^d$*, then* z *is* ε*-essential.*

Definition 4 *The* essential ε-quasi k-medial axis, $\bar{\mathcal{M}}^k(P,\varepsilon)$*, is the set of* ε*-essential points lying on the union of the* i*-dimensional strata of the* ε*-quasi* k*-medial axis over all* $i \leq d-k+1$.

Lemma 4 *For* ε *smaller than the diameter of* P*, the* i*-dimensional stratum of the* ε*-quasi* k*-medial axis has a* i*-dimensional volume bounded by a constant, that does not depend on* ε.

4 Covering Delaunay spheres

The goal of this section is to prove that the intersection of a p-dimensional polyhedron P with any Delaunay sphere Σ is contained in the cover of some point z on the essential ε-quasi k-medial axis, for $k = \lceil \frac{d+1}{p+1} \rceil$. We first state crucial properties of Delaunay spheres and polyhedra before defining the cover of a point. The first property is induced by our sampling condition.

Definition 5 *We say that a sphere* Σ *with center* z *is* ε-almost P-empty *if* $\Sigma \subseteq A(z,\varepsilon)$.

Lemma 5 *Delaunay spheres are* ε*-almost* P*-empty.*

The second property concerns polyhedra.

Definition 6 *We say that a polyhedron* P *is* k-reductible *if for any collection of* $k-1$ *faces* $\{F_1,\ldots,F_{k-1}\}$ *of* P*, there exists a hyperplane that contains the union* $\bigcup_{i=1}^{k-1} F_i$.

Lemma 6 *Any* p*-dimensional polyhedron of* $\mathbb{R}^d$ *is* $\lceil \frac{d+1}{p+1} \rceil$*-reductible.*

We now define the cover of a point $z \in \mathbb{R}^d$. Writing $\pi_x(z)$ for the orthogonal projection of z onto the tangent plane of $x \in P$, we say that x is a critical point of the distance-to-z function if $\pi_x(z) = x$. We define $\chi(z,\varepsilon)$ as the set of critical points lying in $P \cap A(z,\varepsilon)$ and the cover of z as:

$$\text{Cover}(z,\varepsilon) = \bigcup_{x \in \chi(z,\varepsilon)} B(x, 5d\varepsilon).$$

Lemma 7 *Consider a* k*-reductible polyhedron* P *that spans* $\mathbb{R}^d$*. For every* ε*-almost* P*-empty sphere* Σ*, there exists a point* $z \in \bar{\mathcal{M}}^k(P,\varepsilon)$ *such that*

$$\Sigma \cap P \subseteq \text{Cover}(z,\varepsilon).$$

In the next section, it will be convenient to use a slightly different notion of cover. Let $\Pi(z)$ be the set of orthogonal projections of z onto the planes supporting faces of P. We define the extended cover of point z as

$$\text{ExtendedCover}(z,\varepsilon) = \bigcup_{x \in \Pi(z)} B(x, 6d\varepsilon).$$

Lemma 8 *For every points* z *and* z' *with* $\|z - z'\| \leq \varepsilon$*:*

$$\text{Cover}(z,\varepsilon) \subseteq \text{ExtendedCover}(z',\varepsilon).$$

5 Size of Delaunay triangulation

In this section, we collect results from previous sections and establish our upper bound on the number of Delaunay simplices. We then prove that our bound is tight. We recall that the number of points in a λ-sparse ε-sample S of a p-dimensional polyhedron P is $n = \Theta(\epsilon^{-p})$ and that the i-faces of P have $\Theta(\varepsilon^{-i})$ points of S [1].

5.1 Upper bound

Without loss of generality, we may assume that the polyhedron P spans $\mathbb{R}^d$. An ε-sample of the essential ε-quasi k-medial axis is a subset $M \subseteq \bar{\mathcal{M}}^k(P,\varepsilon)$ such that every point $x \in \bar{\mathcal{M}}^k(P,\varepsilon)$ is at distance no more than ε to a point $z \in M$, $\|x - z\| \leq \varepsilon$. We claim that we can construct an ε-sample M of $\bar{\mathcal{M}}^k(P,\varepsilon)$ in such

a way that the i-dimensional stratum of the essential ε-quasi k-medial axis receives $O(\varepsilon^{-i})$ points of M and the number of points in M is $m = O(\varepsilon^{-(d-k+1)})$. This is a consequence of Lemma 4 which says that the i-dimensional volume of the i-dimensional stratum of $\bar{\mathcal{M}}^k(P, \varepsilon)$ is bounded by a constant that does not depend on ε. To establish our upper bound, we map each Delaunay simplex $\sigma \in \mathrm{Del}(S)$ to a point $z \in M$. Consider a Delaunay sphere Σ passing through the vertices of σ. By Lemma 5, Delaunay spheres are ε-almost P-empty. We can therefore combine Lemma 6, Lemma 7 and Lemma 8 and get that for $d \geq 2$ and $k = \lceil \frac{d+1}{p+1} \rceil$, there exists a point $z \in M$ such that

$$\Sigma \cap P \subseteq \mathrm{ExtendedCover}(z, \varepsilon)$$

The extended cover of z is a union of d-balls of radius $6d\varepsilon$, one for each face of the polyhedron and therefore, it contains a constant number of points of S. It follows that the number of simplices that we can form by picking points in the extended cover of z is constant. Hence, each point $z \in M$ is charged with a constant number of Delaunay simplices and using $n = \Omega(\varepsilon^{-p})$, we get that the number of Delaunay simplices is

$$O(m) = O(\varepsilon^{-(d-k+1)}) = O(n^{\frac{d-k+1}{p}}),$$

where $k = \lceil \frac{d+1}{p+1} \rceil$.

5.2 The bound is tight

We now prove that our upper bound is tight. Consider a set of $d+1$ affinely independent points that we partition into $k = \lceil \frac{d+1}{p+1} \rceil$ groups $Q_1, \ldots, Q_k$ in such a way that the maximum number of points in Q_i, over all $i \in [1, k]$, is $p + 1$. Writing q_i for the dimension of the affine space spanned by Q_i, we have

$$\sum_{i=1}^{k} q_i = d - k + 1. \tag{1}$$

Letting C_i be the convex hull of Q_i, we consider the polyhedron $P = \bigcup_{i=1}^{k} C_i$. Let S be a λ-sparse ε-sample of P. The simplex $\sigma = \{s_1, \ldots, s_k\}$ obtained by picking a sample point $s_i \in S \cap C_i$ for $1 \leq i \leq k$ belongs to the Delaunay triangulation. Indeed, since the points $s_1, \ldots, s_k$ are affinely independent, there exists a $(d-1)$-sphere Σ tangent to P at s_i for $1 \leq i \leq k$, whose center lies on the 0-quasi k-medial axis of P. By construction, this sphere encloses no sample point of S in its interior, showing that σ is a Delaunay simplex. Since C_i contains $\Omega(\varepsilon^{-q_i})$ points of S, the amount of Delaunay simplices that we can construct this way is at least

$$\Omega(\varepsilon^{-q_1} \times \cdots \times \varepsilon^{-q_k}) = \Omega(\varepsilon^{-(d-k+1)}) = \Omega(n^{\frac{d-k+1}{p}}).$$

6 Conclusion

This paper answers only the first of many possible questions about the complexity of the Delaunay triangulations of points distributed nearly uniformly on manifolds. Similar bounds for smooth surfaces rather than polyhedra would be of more practical interest. The proof in this paper seems to rely on some properties specific to polyhedra, particularly that sample points on k faces are needed to form a simplex. On the other hand, the tight bound seems to be "right", at least in the sense that it agrees with the well-known bounds in the cases $p = 1$ and $p = d$.

Acknowledgments

The work of the first author was supported by NSF awards CCF–0331736 and CCF–0635250. The work by the second author was supported by CNRS under grant PICS-3416.

References

[1] N. Amenta, D. Attali, and O. Devillers. Complexity of Delaunay triangulation for points on lower-dimensional polyhedra. In *Proc. ACM-SIAM Sympos. Discrete Algorithms (SODA)*, New-Orleans, Lousiana, USA, January 7–9 2007.

[2] N. Amenta, D. Attali, and O. Devillers. A tight bound for the Delaunay triangulation of points on a polyhedron. Technical report, INRIA, 2008.

[3] D. Attali and J.-D. Boissonnat. A linear bound on the complexity of the Delaunay triangulation of points on polyhedral surfaces. *Discrete and Computational Geometry*, 31(3):369–384, 2004.

[4] D. Attali, J.-D. Boissonnat, and A. Lieutier. Complexity of the Delaunay triangulation of points on surfaces: the smooth case. In *Proc. of the 19th ACM Symposium on Computational Geometry*, pages 201–210, 2003.

[5] J. Erickson. Nice point sets can have nasty Delaunay triangulations. *Discrete and Computational Geometry*, 30:109–132, 2003.

[6] M. Golin and H.-S. Na. On the average complexity of 3d-Voronoi diagrams of random points on convex polytopes. *Computational Geometry: Theory and Applications*, 25:197–231, 2003.

[7] M. J. Golin and H.-S. Na. The probabilistic complexity of the Voronoi diagram of points on a polyhedron. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, pages 209–216, 2002.

[8] Gunnells. Stratified spaces twigs. TWIGS talks. `http://www.math.umass.edu/%7Egunnells/talks/singspc.pdf`.

[9] P. McMullen. The maximum number of faces of a convex polytope. *Mathematika*, 17:179–184, 1970.

Discrete Voronoi Diagrams on Surface Triangulations and a Sampling Condition for Topological Guarantee *

Masaki Moriguchi † Kokichi Sugihara †

Abstract

Given a triangulation T of a surface S other than the sphere and a subset P of its vertex set $V(T)$, we define the vertex-based discrete Voronoi diagram $\mathrm{Vor}(P)$ using the shortest path distance on the unweighted graph $G(T)$. In this paper, we provide a sampling condition on P such that $\mathrm{Vor}(P)$ is 3-representative (i.e. any noncontractible cycle in T intersects at least three Voronoi regions). We also show that a 3-representative discrete Voronoi diagram can be easily modified to a decomposition of S such that its dual decomposition is a triangulation of S.

1 Introduction

A triangulation of a surface S (connected, compact 2-manifold without boundary) is a simple graph embedded on S, such that each face is homeomorphic to a 2-cell and is bounded by three edges, and any two faces share at most one edge. For a triangulation T, we denote the vertex set, edge set and face set by $V(T)$, $E(T)$ and $F(T)$, respectively. The graph $(V(T), E(T))$ is denoted by $G(T)$. We refer to [6] for embeddings of graphs into surfaces and to [7] for triangulations in topological graph theory.

Given a triangulation T of a surface S other than the (topological) sphere and a subset P of its vertex set $V(T)$, we define the vertex-based discrete Voronoi diagram $\mathrm{Vor}(P)$ using the shortest path distance on the unweighted graph $G(T)$. (The formal definition is described in Section 2.) In some applications, such as mesh simplification, topology preservation is important [3, 4] and it requires that the dual of $\mathrm{Vor}(P)$ is a triangulation of S. We say that a discrete Voronoi diagram is topology-preserving if its dual is a triangulation of S. This is equivalent to the condition that each Voronoi region is homeomorphic to a disk and the intersection of any two Voronoi regions is either empty, a single vertex or a single edge. We also say that a decomposition of a surface is topology-preserving if its dual is a triangulation of the surface.

*This work is partly supported by the Grant-in-Aid for Scientific Research (S) of the Japan Society for Promotion of Science.

†Department of Mathematical Informatics, University of Tokyo, {Masaki_Moriguchi, sugihara}@mist.i.u-tokyo.ac.jp

If a sample set P is too sparse, the dual of the discrete Voronoi diagram $\mathrm{Vor}(P)$ might not be a triangulation or even the dual might not be defined. Leibon and Letscher study geodesic Voronoi diagrams on Riemannian surfaces and show a sampling condition such that its dual is a triangulation of S [5]. We will show a similar sampling condition in a discrete setting.

In this paper, we provide a sampling condition on a sample set P such that $\mathrm{Vor}(P)$ is 3-representative (i.e. any noncontractible cycle in T intersects at least three Voronoi regions). We also show that a 3-representative discrete Voronoi diagram can be easily modified to a decomposition of S such that its dual decomposition is a triangulation of S.

1.1 Related work

Leibon and Letscher show a sampling condition for geodesic Voronoi diagrams on Riemannian surfaces [5].

Theorem *Let S be a Riemannian surface and P be a set of points on S. Suppose that P satisfies that for every point $x \in S$ there exists a point $p \in P$ such that $d(x,p) \leq \epsilon\, f(x)$, for $\epsilon \leq 0.2$. Then the geodesic Voronoi diagram of P is topology-preserving. Here, $d(x,y)$ is the geodesic distance between x and y, and $f(x)$ is a strong convexity radius of S at x.*

Although the context is different, in surface reconstruction, a sampling condition for restricted Voronoi diagrams on surfaces embedded in $\mathbb{R}^3$ is provided [1, 2]. The restricted Voronoi diagram is the decomposition of the surface constructed by restricting the three-dimensional Voronoi diagram to the surface.

Theorem *Let S be a surface embedded in $\mathbb{R}^3$ and P be a set of points on S. Suppose that P satisfies that for every point $x \in S$ there exists a point $p \in P$ such that $d(x,p) \leq \epsilon f(x)$, for $\epsilon \leq 0.18$. Then the restricted Voronoi diagram of P is topology-preserving. Here, $d(x,y)$ is the Euclidean distance between x and y, and $f(x)$ is a local feature size of S at x.*

While they derive sampling conditions for topology-preserving continuous Voronoi diagrams, we derive a sampling condition for 3-representative discrete Voronoi diagrams.

2 Preliminaries

In this section, we first formally define the discrete Voronoi diagram. Then we provide basic lemmas including the 3-path condition for noncontractible cycles.

2.1 Discrete Voronoi diagrams

We define discrete Voronoi diagrams on a triangulation T using the shortest path distance on the unweighted graph $G(T)$. The length of a path is the number of edges in the path, and the distance $d(x, y)$ between two vertices x and y is the length of the shortest path connecting x and y.

Let T be a triangulation on a surface S and $P = \{p_i\}_{i=1}^{k} \subset V(T)$ be a set of vertices. We first define the Voronoi set V_i for each sample vertex p_i as

$$V_i = \{v \in V(T) \mid d(v, p_i) \leq d(v, p_j),\ p_j \neq p_i\}\ .$$

We also require that each Voronoi set V_i satisfies the following condition:

$$d_{G[V_i]}(v, p_i) = d(v, p_i) \quad (\forall v \in V_i)\ ,$$

where $d_{G[V_i]}(x, y)$ is the length of the shortest path in $G[V_i]$, which is the subgraph of $G(T)$ induced by V_i, between x and y. If some vertex has more than one closest vertices in P, it must be assigned to exactly one Voronoi set using some tie-breaking rule. One can use arbitrary tie-breaking rules as long as the condition on the Voronoi sets is satisfied. The Voronoi sets can computed, for example, using the multiple-source Dijkstra algorithm.

Then, we define the discrete Voronoi diagram $\mathrm{Vor}(P)$ of P as a decomposition of S into subsets, called Voronoi regions. The Voronoi region V_i^* of a sample vertex p_i is defined as the set of dual faces associated with V_i:

$$V_i^* = \{v^* \in V^*(T) \mid v \in V_i\}\ ,$$

where v is the primal vertex of a dual face v^* and $V^*(T)$ is the dual faces of T. Due to the condition of the Voronoi sets, each Voronoi region is connected.

2.2 Basic lemmas

Let T be a triangulation on a surface S other than the sphere. A discrete Voronoi diagram is said to be 3-representative if any noncontractible closed curve on S intersects at least three Voronoi regions. A noncontractible closed curve is a closed curve which cannot be continuously deformed into a single point. However, we do not need to consider all noncontractible closed curves on S; it is enough to consider only noncontractible cycles in T, since T is a triangulation. Thus, we obtain the following lemma.

Lemma 1 *Let T be a triangulation on a surface other than the surface. A discrete Voronoi diagram on T is 3-representative if and only if any noncontractible cycle in T intersects at least three Voronoi regions.*

Next, we introduce the 3-path condition for noncontractible cycles. Let $\mathcal{K}$ be a family of cycles. Let u, v be vertices of T and P_1, P_2, P_3 be internally disjoint paths connecting u and v. $\mathcal{K}$ is said to satisfy the 3-path condition if it satisfies the following. If one of the three cycles $P_1 \cup P_2$, $P_1 \cup P_3$ and $P_2 \cup P_3$ is contained in $\mathcal{K}$, then at least one of the rest of cycles is contained in $\mathcal{K}$.

Thomassen showed that noncontractible cycles satisfy the 3-path condition [8].

Lemma 2 *Let u, v be vertices of T and P_1, P_2, P_3 be internally disjoint paths connecting u and v. If one of the three cycles $P_1 \cup P_2$, $P_1 \cup P_3$ and $P_2 \cup P_3$ is noncontractible, then at least one of the rest of cycles is also noncontractible.*

From this lemma, we can prove that if a cycle C, which can be decomposed into a set of cycles $\{K_i\}$, is noncontractible then at least one of K_i is noncontractible. We use this corollary to prove our sampling condition.

3 Modification for topology preservation

We show that the 3-representative discrete Voronoi diagram can be easily modified to be topology-preserving. A discrete Voronoi diagram on a surface triangulation is said to be topology-preserving if and only if its dual is a triangulation of the surface. The following lemma implies the importance of the condition to be 3-representative.

Lemma 3 *Let T be a triangulation on a surface S other than the sphere and $P \subset V(T)$ be a set of vertices. If $\mathrm{Vor}(P)$ is 3-representative, it can be modified such that its dual is a triangulation of S.*

Proof. We show that a 3-representative decomposition can be modified to be topology-preserving. Suppose that $\mathcal{D}$ be a 3-representative decomposition. The dual of $\mathcal{D}$ is a 3-representative triangular embedding. By repeatedly applying edge flips to this triangular embedding, we get a triangulation. We apply the dual operations of these edge flips to $\mathcal{C}$, and we get a topology-preserving decomposition. □

4 Sampling condition

In this section, we provide a sampling condition on a sampling set so that its discrete Voronoi diagram is 3-representative.

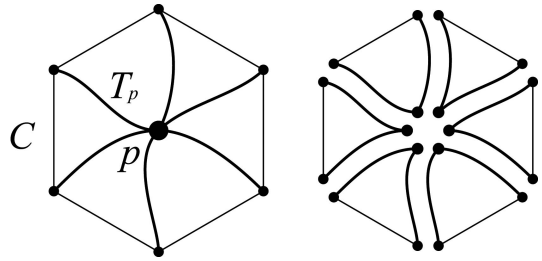


Figure 1: Left: all the vertices in a noncontractible cycle C is contained in the Voronoi region of p. Right: cycle C can be decomposed into a set of cycles.

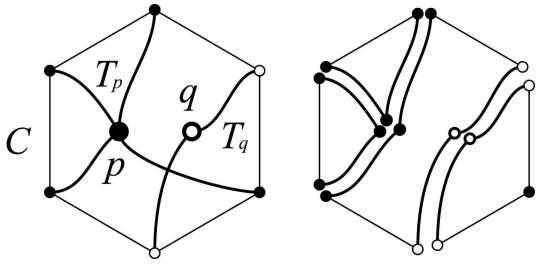


Figure 2: Left: all the vertices in a noncontractible cycle C is contained in the Voronoi region of p and the Voronoi region of q. Right: cycle C can be decomposed into a set of cycles.

We first define a function $D(l)$ ($l \geq 3$) as follows. Suppose that C_l is a cycle of length l and u, v are two vertices on C_l. Let $d_{u,v}$ denote the maximum distance between x and $\{u, v\}$, taken over all $x \in C_l$. Then we define $D(l)$ to be the minimum of $d_{u,v}$ taken over all $u, v \in C_l$,

$$\begin{aligned} D(l) &= \min_{u,v \in C_l} d_{u,v} \\ &= \min_{u,v \in C_l} \max_{x \in C_l} d(x, \{u, v\}) \\ &= \min_{p+q=l, p \geq 0, q \geq 0} \left\lfloor \frac{\max\{p, q\}}{2} \right\rfloor \\ &= \min_{\lceil l/2 \rceil \leq p \leq l} \left\lfloor \frac{p}{2} \right\rfloor \\ &= \left\lfloor \frac{l+1}{4} \right\rfloor . \end{aligned}$$

Noting that $d < D(l) \iff d \leq D(l) - 1$, we define another function

$$\delta(l) = D(l) - 1 = \left\lfloor \frac{l-3}{4} \right\rfloor .$$

Then, we present a sampling condition using this function.

Lemma 4 *Let T be a triangulation on a surface other than the sphere and P be a subset of its vertex set. Suppose that P satisfies that, for every noncontractible cycle C in T, each vertex $v \in C$ has a vertex $p \in P$ within distance $\delta(\ell(C))$, where $\ell(C)$ is the length of C. Then the discrete Voronoi diagram of P is 3-representative.*

Proof. We will show any noncontractible cycle in T intersect at least three Voronoi regions by contradiction. Suppose that a noncontractible cycle C intersects only one Voronoi region of a vertex $p \in P$. Let T_p be a shortest-path tree (actually it is a breadth-first-search tree) rooted at p. T_p decomposes C into a set $\mathcal{K}$ of cycles, see Figure 1. Some of $\mathcal{K}$ might be non-simple. In such case, removing repeated edges within each cycle resolves non-simple cycles. Then by the 3-path condition, at least one of these cycles is a noncontractible cycle. Let K be such a cycle. See Figure 3 (left) for the notations. We can assume $s_1 \geq s_2$ without loss of generality. Then, we will show that the sampling condition for K contradicts the assumption that the distance from u_1 to its closest vertex in P is s_1. By the sampling condition for K, u_1 has a vertex in P within distance

$$\delta(s_1 + s_2 + 1) = \left\lfloor \frac{s_1 + s_2 - 2}{4} \right\rfloor < s_1 .$$

This contradicts the assumption that the distance from u_1 to its closest vertex in P is s_1.

Next, suppose that a noncontractible cycle C intersects only two Voronoi regions of vertices $p \in P$ and $q \in P$. Let T_p and T_q be shortest-path trees rooted at p and q, respectively. T_p and T_q decomposes C into a set $\mathcal{K}$ of cycles, see Figure 2. Some of $\mathcal{K}$ might be non-simple. In such case, removing repeated edges within each cycle resolves non-simple cycles. Then by the 3-path condition, at least one of these cycles is a noncontractible cycle. Let K be such a cycle. There are three configurations for K, see Figure 3. For each case, the sampling condition for K contradicts the assumption that the distance from u_1 to its closest vertex in P is s_1.

Case 1 (Figure 3 left): We can assume $s_1 \geq s_2$ without loss of generality. By the sampling condition for the new noncontractible cycle, u_1 has a vertex in P within distance

$$\delta(s_1 + s_2 + 1) = \left\lfloor \frac{s_1 + s_2 - 2}{4} \right\rfloor < s_1 .$$

This contradicts the assumption that the distance from u_1 to its closest vertex in P is s_1.

Case 2 (Figure 3 center): We can assume $s_1 \geq s_2$ without loss of generality. By the sampling condition for the new noncontractible cycle, u_1 has a vertex in P within distance

$$\delta(s_1 + s_2 + 2) = \left\lfloor \frac{s_1 + s_2 - 1}{4} \right\rfloor < s_1 .$$

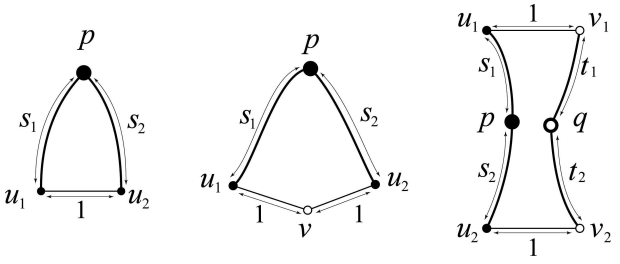


Figure 3: Left: Case 1. A cycle consists of two paths in the shortest-path tree T_p and an edge u_1u_2 in C. Center: Case 2. A cycle consists of two paths in the shortest-path tree T_p and two edges u_1v and u_2v in C. Right: Case 3. A cycle consists of two paths in the shortest-path tree T_p, two paths in the shortest-path tree T_q, and two edges u_1v_1 and u_2v_2 in C.

This contradicts the assumption that the distance from u_1 to its closest vertex in P is s_1.

Case 3 (Figure 3 right): We can assume $s_1 \geq s_2, t_1, t_2$ without loss of generality. By the sampling condition for the new noncontractible cycle, u_1 has a vertex in P within distance

$$\delta(s_1+s_2+t_1+t_2+2) = \left\lfloor \frac{s_1+s_2+t_1+t_2-1}{4} \right\rfloor < s_1 .$$

This contradicts the assumption that the distance from u_1 to its closest vertex in P is s_1.

□

In this sampling condition, the number of conditions is too large and they are verbose. To simplify the sampling condition, we define the edge-width, denoted by $ew(v)$, at a vertex v as the minimum length of a noncontractible cycle containing v. Using this quantity, the sampling condition can be simplified into the following theorem.

Theorem 5 *Let T be a triangulation on a surface other than the sphere and P be a subset of its vertex set. Suppose that P satisfies that for every vertex $v \in V(T)$ there exists a vertex $p \in P$ such that $d(v,p) \leq \left\lfloor \frac{ew(v)-3}{4} \right\rfloor$. Then the discrete Voronoi diagram of P is 3-representative.*

The edge-width at a vertex v measures the size of a topological feature at v, and this theorem states that if a sample set is dense enough with respect to the edge-width then its discrete Voronoi diagram is 3-representative. Note that our definition of the edge-width is slightly different than that defined in topological graph theory where the edge-width is defined for embedding of graphs and it is the minimum length of a noncontractible cycle in the embedding [6].

If a set of vertices P satisfies the assumption in Theorem 5, then the discrete Voronoi diagram of P is 3-representative and, as shown in Section 3, we can get a topology-preserving decomposition by modifying $\mathrm{Vor}(P)$.

5 Discussion

We have presented a sampling condition for 3-representative discrete Voronoi diagrams. We use the shortest path distance on the unweighted graph and the sampling density is measured using edge-width. Since the shortest path distance on the weighted graph is much more desirable, we would like to deal with it. Currently we are considering weighted graphs whose weights satisfy the triangle inequality.

While we study vertex-based discrete Voronoi diagrams on a surface triangulation, face-based discrete Voronoi diagrams are also used in geometry processing [3]. By duality, they can be treated as vertex-based discrete Voronoi diagrams on the dual embedding of the triangulation. By defining the face-width at a vertex v as the minimum number of faces whose union contains a noncontractible cycle containing v, we can derive a similar sampling condition using face-width instead of edge-width. However, contrary to triangulations, 3-representative Voronoi diagrams cannot necessarily be modified to polyhedral embeddings. We would also like to resolve this issue.

References

[1] Amenta, N. and Bern, M. Surface reconstruction by Voronoi filtering. *Discrete and Computational Geometry*, **22** (4): 481–504, 1999.

[2] Dey, T. K. *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis*. Cambridge University Press, 2006.

[3] Eck, M., DeRose, T., Duchamp, T., Hoppe, H., Lounsbery, M., and Stuetzle, W. Multiresolution analysis of arbitrary meshes. In *Proceedings of SIGGRAPH*, 173–182, 1995.

[4] Klein, A., Certain, A., DeRose, A., Duchamp, T., and Stuetzle, W. Vertex-based Delaunay triangulation of meshes of arbitrary topological type. Technical Report, University of Washington, 1997.

[5] Leibon, G. and Letscher, D. Delaunay triangulations and Voronoi diagrams for Riemannian manifolds. In *Proceedings of Symposium on Computational Geometry*, 341–349, 2000.

[6] Mohar, B. and Thomassen, C. *Graphs on Surfaces*. Johns Hopkins University Press, 2001.

[7] Negami, S. Triangulations. In Gross, J. L. and Yellen, J. eds., *Handbook of Graph Theory*, CRC Press, 737–760, 2004.

[8] Thomassen, C. Embeddings of graphs with no short noncontractible cycles. *Journal of Combinatorial Theory, Series B*, **48** (2): 155–177, 1990.

On the Locality of Extracting a 2-Manifold in $\mathbb{R}^3$

Daniel Dumitriu* Stefan Funke† Martin Kutz* Nikola Milosavljevic‡

Abstract

Algorithms for reconstructing a 2-manifold from a point sample in $\mathbb{R}^3$ based on Voronoi-filtering like *CRUST* [1] or *CoCone* [2] still require – after identifying a set of candidate triangles – a so-called *manifold extraction* step which identifies a subset of the candidate triangles to form the final reconstruction surface. Non-locality of the latter step is caused by so-called *slivers* – configurations of 4 almost cocircular points having an empty circumsphere with center close to the manifold surface.

We show that under a certain mild condition – local uniformity – which typically holds in practice but can also be enforced theoretically, one can compute a reconstruction using an algorithm whose decisions about the adjacencies of a point only depend on nearby points. While the theoretical proof requires an extremely high sampling density, our prototype implementation – which is described in a companion paper [5] – exhibits pretty good results on typical sample sets and might have some potential in particular in parallel computing or external memory scenarios due to its local mode of computation. The full version of this paper is available under [4].

1 Introduction

Reconstructing a surface Γ in $\mathbb{R}^3$ from a finite point sample V has attracted a lot of attention both in the computer graphics community as well as in the computational geometry community. While in the former the emphasis is mostly on algorithms that work 'well in practice', the latter has focused on algorithms that come with a theoretical guarantee: if the point sample V satisfies a certain *sampling condition*, the output of the respective algorithm is guaranteed to be 'close' to the original surface.

In [1], Amenta and Bern proposed a framework for rigorously analyzing algorithms reconstructing smooth closed surfaces. They define for every point $p \in \Gamma$ on the surface the *local feature size* $\mathrm{lfs}(p)$ as the distance of p to the *medial axis*[1] of Γ. A set of points $V \subset \Gamma$ is called a ε-sample of Γ if $\forall p \in \Gamma\ \exists s \in V : |sp| \leq \varepsilon \cdot \mathrm{lfs}(p)$. For sufficiently small ε, Amenta and Bern define a *canonical correct reconstruction* of V with respect to Γ as the set of Delaunay triangles that are dual to Voronoi edges in the Voronoi diagram of V that are intersected by the surface Γ. Unfortunately, due to certain point configurations called *slivers* – 4 (almost) cocircular points that are nearby on the surface and have an empty, (almost) diametral circumsphere – it is not possible to algorithmically determine the canonical correct reconstruction of V without knowing Γ. Algorithms have been proposed, though, that determine a collection of Delaunay triangles which form a piecewise linear surface that is topologically equivalent to the canonical correct reconstruction and converges to the latter both point-wise as well as in terms of the surface normals as the sampling density goes to infinity ($\varepsilon \to 0$).

The CoCone algorithm [2] is one example; in its last step, it first removes triangles with free edges and then determines the final reconstruction as the outside surface of the largest connected component of the remaining triangles; observe that this is a highly non-local operation. There have been attempts to locally decide for each sample p which of the candidate triangles to keep for the final reconstruction; such local decisions might disagree, though, and hence the selected triangles do not patch up to a closed manifold. Again, the reason why local decisions might disagree is the presence of *slivers* which induce a Voronoi vertex inside the CoCone region of the involved sample points. Each involved sample point has to decide whether in 'its opinion' the true surface Γ intersects above or below the Voronoi vertex and create the respective dual Delaunay triangles. If these decisions are not coordinated contradicting decisions are made. Not only in theory but even in practice the manifold extraction step is still quite challenging and requires deliberate engineering to actually work as desired.

One potential way to obtain a local manifold extraction step is to decide on triangles/adjacencies in a conservative manner by only creating those triangles/adjacencies which are 'safe', i.e. where essentially the respective dual Voronoi edge/face completely pierces the CoCone region. It is unclear, though, how much connectivity is lost – whether the resulting graph is connected at all and how big potential holes/faces are. The main contribution of this paper is to show that it is actually possible to make local decisions but still guarantee that the resulting graph exhibits topological equivalence to the original surface. That is, it is connected, locally planar, and

*Max-Planck-Institut für Informatik, Saarbrücken, Germany, dumitriu@mpi-inf.mpg.de

†Ernst-Moritz-Arndt-Universität, Greifswald, Germany, stefan.funke@uni-greifswald.de

‡Stanford University, Stanford, CA, U.S.A., nikolam@stanford.edu

[1] The medial axis of Γ is defined as the set of points which have at least 2 closest points on Γ.

contains no large holes.

In [6] Funke and Milosavljevic present an algorithm for computing *virtual coordinates* for the nodes of a wireless sensor network which are themselves unaware of their location. Their approach crucially depends on a subroutine to identify a provably planar subgraph of a communication graph that is a quasi-unit-disk graph. The same subroutine will also be used in our surface reconstruction algorithm presented in this paper.

While we deal with the problem of slivers in some sense by avoiding or ignoring them, another approach coined *sliver pumping* has been proposed by Cheng et al. in [3]. Their approach works for smooth k-manifolds in arbitrary dimension, though its practicality seems uncertain. There are, of course, other non-Voronoi-filtering-based algorithms for manifold reconstruction which do not have a manifold extraction step; they are not in the focus of this paper, though.

Our contribution

We propose a novel method for extracting a 2-manifold from a point sample in $\mathbb{R}^3$. Our approach fundamentally differs from previous approaches in two respects: first it mainly operates combinatorially on a graph structure, which is derived from the original geometry; secondly, the created adjacencies/edges are "conservative" in a sense that two samples are only connected if there is a safe, sliver-free region around the two samples. Interestingly we can show, though, that conservative edge creation only leads to small, constant-size faces in the respective reconstruction, hence completion to a triangulated piecewise linear surface can easily be accomplished using known techniques. The most notable advantage compared to previous Voronoi-filtering based approaches is that the manifold extraction step can be performed locally, i.e. it only relies on adjacency information of geometrically nearby points.

While the theoretical analysis requires an absurdly high sampling density – like most of the above mentioned algorithms do – our prototype implementation of the novel local manifold extraction step (see companion paper [5]) suggests that the approach is viable even for practical use.

From a technical point of view, two insights are novel in this paper (and not a result of the mere combination of previous results): first, we show that the neighborhood graph that our algorithm constructs is locally a *quasi-unit-disk graph*; it is this property that allows us to actually make use of the machinery developed in [6]. Second, we provide a more elegant and much stronger result about the density of the extracted planar graph based on the β-skeleton and power-spanner properties; this insight also improves the overall result in [6].

2 Graph-based, conservative adjacencies

In this section we present an algorithm that given a ε-sample V from a closed smooth 2-manifold Γ in $\mathbb{R}^3$ computes a faithful reconstruction of V with respect to Γ as a subcomplex of the Delaunay tetrahedralization of V. The outline of our method is as follows (with the novel steps being 2.–5.):

1. Determine a Lipschitz function $\phi(v)$ for every $v \in V$ which lower-bounds $\varepsilon \operatorname{lfs}(v)$ (as in [7])
2. Construct a local neighborhood graph $G(V)$ by creating an edge from every point v to all other points v' with $|vv'| \leq O(\phi(v))$.
3. Compute a subsample S of (V)
4. Identify adjacencies between elements in S based on the connectivity of $G(V)$ (as in [6])
5. Use geometric positions of the points in S to identify faces of the graph induced by certified adjacencies when embedded on the manifold
6. Triangulate all non-triangular faces
7. reinsert points in $V - S$ by computing the weighted Delaunay triangulations on the respective faces (as in [7])

The core components of the correctness proof of this approach are:

- We show that the local neighborhood graph corresponds locally to a quasi-unit-disk graph for a set of points in the plane.
- The identified adjacencies locally form a planar graph.
- This locally planar graph has faces of bounded size.

Essentially this means that we cover Γ by a mesh with vertex set S consisting of small enough cells that the topology of Γ is faithfully captured. Note that the first and last item from above are original and novel to this paper and do not follow from our previous results in [7] and [6] (the last item makes the theoretical result in [6] much stronger).

We first discuss the 2-dimensional case, where we are given a uniform ε-sampling (i.e. the local feature size is 1 everywhere) of a disk and show that steps 2. to 5. yield a planar graph with 'small' faces. Then we show how the same reasoning can be applied to the 3-dimensional case.

The main rationale of our approach is the "conservative" creation of adjacencies; that is, we only create an edge between two samples if in any good reconstruction the two points are adjacent, which can be interpreted as creating edges only in the absence of slivers in the vicinity.

2.1 Conservative adjacencies in $\mathbb{R}^2$

Let V be a set of n points that form a ε-sampling of the disk of radius R around the origin o, that is, $\forall p \in \mathbb{R}^2$ with $|po| \leq R$, $\exists v \in V : |vp| \leq \varepsilon$.

Definition 1 *A graph $G(V, E)$ on V is called a α-quasi-unit-disk-graph (α-qUDG) for $\alpha \in [0, 1]$ if for $p, q \in V$*

- *if $|pq| \leq \alpha$ then $(p, q) \in E$*
- *if $|pq| > 1$ then $(p, q) \notin E$*

That is, for distances between α and 1 the presence of an edge is left open.

Within G we consider the distance function d_G defined by the (unweighted) graph distances in $G(V, E)$. Let $k \geq 1$, we call a set $S \subseteq V$ a *tight k-subsample* of V if

- $\forall s_1, s_2 \in S$: $d_G(s_1, s_2) > k$
- $\forall v \in V$: $\exists s \in S$ with $d_G(v, s) \leq k$.

A tight k-subsample of V can easily be obtained by a greedy algorithm which iteratively selects a so far unremoved node v into S and removes all nodes at distance at most k from consideration.

The following algorithm determines adjacencies between nodes in S based on a *Graph Voronoi diagram* such that the induced graph on S remains planar.

2.1.1 Graph-based conservative adjacencies

The idea for construction and the planarity property of our construction are largely derived from the geometric intuition. The planarity follows from the fact that our constructed graph – we call it *combinatorial Delaunay map* of S, short CDM(S) – is the *dual graph* of a suitably defined partition of the plane into simply connected disjoint regions.

In the following we use the method for identifying adjacencies between nodes in S purely based on the graph connectivity as described in [6]; the reasoning relies on the fact that the original communication graph is not an arbitrary graph but reflects the geometry of the underlying domain by being a qUDG.

First we introduce a *labeling of $G(V, E)$* for a given set $S \subseteq V$ assuming that all elements in V (and hence in S) have unique IDs that are totally ordered.

Definition 2 *Consider a vertex $a \in S$ and a vertex $v \in V - S$. We say that v is an a-vertex (or: labeled with a) if a is one of the elements in S which is closest to v (in graph distance), and a has the smallest ID among such.*

Clearly, this rule assigns unique labels to each vertex and edge, due to the uniqueness of nodes' IDs. Also note that any $a \in S$ is an a-vertex. Next we present a criterion for creating adjacencies between vertices in S.

Definition 3 *Vertices $a, b \in S$ are adjacent in CDM(S) iff there exists a path from a to b whose 1-hop neighborhood (including the path itself) consists only of a- and b-vertices, and such that in the ordering of the nodes on the path (starting with a and ending with b) all a-nodes precede all b-nodes.*

We have the following result of [6]:

Theorem 1 *If G is an α-qUDG with $\alpha \geq \frac{1}{\sqrt{2}}$ and S a tight k-subsample of G, then CDM(S) is a planar graph.*

Of course, just planarity as such is not too hard to guarantee – one could simply return a graph with no edges.

2.1.2 CDM(S) is dense (!)

Interestingly we can show that in spite of the conservative adjacency creation, CDM(S) is relatively dense, in particular it exhibits (internal) faces of size $O(1)$. Due to space restrictions we cannot provide a proof in this paper but instead refer to the complete version under [4], therefore we only state the following corollary:

Corollary 2 *The graph induced by S and the adjacencies identified by our algorithm is planar, connected and has (internal) faces of size $O(1)$.*

2.2 Conservative adjacencies in $\mathbb{R}^3$

All the reasoning which has been concentrating on a flat, planar setting can be translated with little effort to our actual setting in $\mathbb{R}^3$ as we can show that locally the neighborhood graph we construct looks like an α-quasi-unit-disk-graph.

Here the steps of our algorithm are as follows: (1) we have to compute a Lipschitz[2] function ϕ with $\phi(p) \leq \varepsilon \operatorname{lfs}(p)$ for all $p \in V$. This can be done using the procedure given in [7] in near-linear time. Then in step (2) the graph $G(V, E)$ is constructed by creating edges between samples p_1, p_2 iff $|p_1p_2| \leq 6 \cdot \phi(p_1)$ or $|p_1p_2| \leq 6 \cdot \phi(p_2)$ (Note that the constant 6 is somewhat arbitrary and only chosen such that every sample is connected to at least its neighbors in the canonical correct reconstruction). The following steps (3) to (5) are exactly the same as in the 2-dimensional case. We now want to argue that locally around a sample point p the constructed graph looks like an α-quasi-unit-disk graph. The following basic observation are easy to derive:

[2] In fact one computes a δ-approximate ω-Lipschitz function.

Lemma 3 *For $\gamma \leq 1/16$ the $\gamma \operatorname{lfs}(p)$-neighborhood of p in the constructed graph is an α-quasi-unit-disk graph with $\alpha > \frac{1}{\sqrt{2}}$.*

Now we can invoke Corollary 2 which implies that locally for any $p \in S$ the graph constructed by our algorithm is planar, connected and has internal faces of constant size.

What does this mean? The graph that we constructed on the subsample of points S is a *mesh* that is locally planar and covers the whole 2-manifold. The mesh has the nice property that all its *cells* (aka faces) have constant size (that is, number of bounding vertices). The edge lengths of the created adjacencies between S are proportional to the respective local feature sizes. Therefore its connectivity structure faithfully reflects the topology of the underlying 2-manifold.

2.2.1 Algorithm epilog

We did not talk about steps (6) and (7) of our approach since they follow exactly the description in [7] and are not novel to this work; we nevertheless give a brief summary here.

Essentially in step (6) we triangulate non-triangular faces by projecting them into a nearby (almost) tangent plane and computing the Delaunay triangulation. The resulting triangulated faces behave nicely since all faces have small size (the respective points hence are almost coplanar) and because S is a *locally uniform* sampling of the surface. In step (7) the points pruned in step (3) are reinserted by computing a weighted Delaunay triangulation on the supporting planes of the respective faces. The resulting triangulations are guaranteed to patch up.

The proofs for convergence both point-wise as well as with respect to triangle normals can be carried over from [7] since S can be made an arbitrarily good, locally uniform ε'-sampling (the original ε-sampling V has to be accordingly denser, i.e. $\varepsilon \ll \varepsilon'$). Therefore, the same theorem holds for the result of our algorithm:

Theorem 4 *There exists ε^* such that for all $\varepsilon < \varepsilon^*$, smooth surfaces Γ in $\mathbb{R}^3$ and ε-samplings $V \subset \Gamma$, the triangulated surface $\widetilde{\Gamma}$ output by our algorithm satisfies the following conditions:*

1. Bijection*: $\mu : \widetilde{\Gamma} \to \Gamma$, determined by closest point, is a bijection*
2. Pointwise Approximation*: For all $x \in \widetilde{\Gamma}$, $d(x, \mu(x)) = O(\varepsilon^2 \operatorname{lfs}(\mu(x)))$*
3. Normal Approximation*: For all $x \in \widetilde{\Gamma}$, $\angle n_{\widetilde{\Gamma}}(x) n_\Gamma(\mu(x)) = O(\varepsilon)$ where $n_F(y)$ denotes the (outside) normal of F at y[3].*
4. Topological Correctness*: Γ and $\widetilde{\Gamma}$ have the same topological type.*

3 Conclusions

We proposed a novel algorithm for extracting a 2-manifold from a point sample in $\mathbb{R}^3$, differing fundamentally from previous approaches by mainly operating combinatorially on a graph structure derived from the original geometry, and conservatively creating adjacencies/edges. We showed that when local uniformity is satisfied, the algorithm computes a faithful reconstruction of the original surface, both point-wise and in terms of triangle normals; this could also be observed with our prototype implementation, which is described in [5].

Theoretically our approach has the potential to work for reconstructing 2-manifolds even in higher dimensions. It does not extend to non-2-manifolds, though, as the "local planarity property" of a graph that our algorithm crucially depends upon, has no equivalent for non-2-manifolds.

While we proved theoretically that our new method allows for a localized manifold extraction step when reconstructing a 2-dimensional manifold in $\mathbb{R}^3$, it remains to be seen whether this localization property leads to practically more effcient algorithms in the parallel computing or external memory scenario.

References

[1] N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. In *Proc. 14th ACM SoCG*, 1998.

[2] N. Amenta, S. Choi, T. K. Dey, and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. In *Proc. 16th SoCG*, 2000.

[3] S.-W. Cheng, T.K. Dey, and E.A. Ramos. Manifold reconstruction from point samples. In *ACM-SIAM SODA*, pages 1018–1027, 2005.

[4] D. Dumitriu, S. Funke, M. Kutz, and N. Milosavljevic. On the locality of reconstructing a 2-manifold in $\mathbb{R}^3$. manuscript. http://www.mpi-inf.mpg.de/∼funke/Papers/LocalRecon.pdf.

[5] D. Dumitriu, S. Funke, M. Kutz, and N Milosavljevic. How much Geometry it takes to Reconstruct a 2-Manifold in $\mathbb{R}^3$. In *Proc. 10th ACM-SIAM ALENEX*, pages 65–74, 2008.

[6] S. Funke and N. Milosavljevic. Network Sketching or: "How Much Geometry Hides in Connectivity? – Part II". In *Proc. ACM-SIAM SODA*, pages 958–967, 2007.

[7] S. Funke and E. Ramos. Smooth-surface reconstruction in near-linear time. In *Proc. ACM-SIAM SODA*, 2002.

[3] For $\widetilde{\Gamma}$ the normal is well-defined in the interior of triangles; at edges and vertices it can be defined as an interpolation from that at the incident triangles.

Arrangements on Surfaces of Genus One: Tori and Dupin Cyclides

Eric Berberich* Michael Kerber*

Abstract

An algorithm is presented to compute the exact arrangement induced by arbitrary algebraic surfaces on a parametrized ring Dupin cyclide, including the special case of the torus. The intersection of an algebraic surface of degree n with a reference cyclide is represented as a real algebraic curve of bi-degree $(2n, 2n)$ in the cyclide's two-dimensional parameter space. We use Eigenwillig and Kerber [11] to compute a planar arrangement of such curves and extend their approach to obtain more asymptotic information about curves approaching the boundary of the cyclide's parameter space. With that, we can base our implementation on a general software framework by Berberich et. al. [3] to construct the arrangement on the cyclide. Our contribution provides the demanded techniques to model the special topology of the reference surface of genus one. Our experiments show no combinatorial overhead of the framework, i.e., the overall performance is strongly coupled to the efficiency of the implementation for arrangements of algebraic plane curves.

1 Introduction

Consider a surface S in $\mathbb{R}^3$ and a set C of curves on S. The arrangement $\mathcal{A}(C)$ is the subdivision of S into cells of dimensions zero, one, and two with respect to C. The cells are called vertices, edges, and faces, respectively.

Berberich et al. [3] introduced a general software framework for sweeping a set of curves on a parametric surface S. We present an implementation for the case that S is a ring Dupin cyclide and the arrangement on it is induced by intersections of S with algebraic surfaces of arbitrary degree. Our approach always computes the exact arrangement, undistorted by rounding errors, of the given input. It also handles all degeneracies like singular points or intersections with high multiplicity.

Dupin Cyclides have been introduced by Dupin [9] as surfaces whose lines of curvature are all circular. One can think of a (ring) Dupin cyclide as a torus with variable tube radius. Dupin cyclides are the generalization of the "natural" geometric surfaces like planes, cylinders, cones, spheres and tori, what makes them useful for applications in solid modeling; compare, e.g., [6], [15], [16], [8].

Our algorithm is this: we follow the framework of [3], and perform a sweep-line algorithm [2] on the intersection curves of the Dupin cyclide with the surfaces in the parameter space. The primitives of the sweep are specified by a model of the *GeometryTraits* concept which is given by the recent work of Eigenwillig and Kerber [11]. With that model, one can sweep over algebraic plane curves of arbitrary degree. The applied sweep line algorithm interacts with a model of the *TopologyTraits* concept; this model controls the creation and manipulation of arrangement features at the boundary of the parameter space, i.e., *identifications* in our case. We implemented such a model for the case of a Dupin cyclide. The arrangement on the Dupin cyclide is represented by a doubly-connected edge-list (DCEL), where points are attached to vertices and curves are stored with edges. Our implementation in C++ deeply benefits from generic programming capabilities, i.e., we are using CGAL's[1] class template `Arrangement_on_surface_2` that expects proper models of the `GeometryTraits` and the `TopologyTraits` concept.

Related work: Arrangements in the plane have been well studied during the past decades, and also quite a number of exact and efficient implementations appeared [13]. Two-dimensional arrangements on surfaces, especially with exact implementation, became more popular recently, e.g., arrangements of great arcs on a sphere [3], arrangements of small arcs on a sphere by Cazals and Loriot [7]. The most complicated surfaces considered so far are arrangements induced by quadrics intersecting a reference quadric. Three approaches exist. The first actually computes more, namely the adjacency relationship between intersection of a set of quadrics [10]. The other two project the intersection curves onto the xy-plane. The original work [4] maintains two arrangements, one for the lower part of the reference quadric and one for its upper part; a connection between them is missing. Instead, [3] introduces a small extension of the projection to simulate the parameter space of the reference quadric. This way, it benefits from the framework that we also apply for ring cyclides. Instead of such a simulation, the sweep on a cyclide is explicitly performed in parameter space.

A more detailed version of this paper appears in [5].

*Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany, email: {eric,mkerber} @mpi-inf.mpg.de

[1] See the project homepage: http://www.cgal.org

2 Dupin cyclides

The maybe most intuitive way of constructing a Dupin cyclide (called *cyclide* for brevity) goes back to Maxwell, we cite it from Boehm [6]:

> Let a sufficiently long string be fastened at one end to one focus f of an ellipse, let the string be kept always tight while sliding smoothly over the ellipse, then the other end z sweeps out the whole surface of a cyclide Z.

Note that choosing a circle in this construction yields a torus. We assume that the cyclide is in *standard position and orientation*, i.e., the chosen base ellipse is given by $(x/a)^2 + (y/b)^2 = 1, \quad a \geq b > 0$.

We define $c := \sqrt{a^2 - b^2}$, and μ as the length of the string minus a. We assume that $c < \mu \leq a$ which means that the cyclide has no self-intersections (it is a ring cyclide).

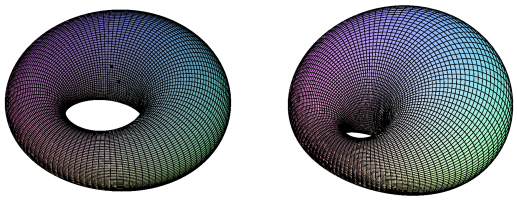


Figure 2.1: (Left) Cyclide with $a = 1$, $b = 0.99$, $\mu = 0.5$, (Right) Cyclide with a=13, $b = 12$, $\mu = 9$

The parameterization of the cyclide [14] is given by

$$\begin{pmatrix} \phi \\ \psi \end{pmatrix} \mapsto \begin{pmatrix} \frac{\mu(c - a\cos\phi\cos\psi) + b^2\cos\phi}{a - c\cos\phi\cos\psi} \\ \frac{b(a - \mu\cos\psi)\sin\phi}{a - c\cos\phi\cos\psi} \\ \frac{b(c\cos\phi - \mu)\sin\psi}{a - c\cos\phi\cos\psi} \end{pmatrix}$$

with $\phi, \psi \in [-\pi, \pi]$. For $\phi = \pi$ and $\phi = -\pi$, this yields the *tube circle* $(x + a)^2 + z^2 = (\mu + c)^2$ in the plane $y = 0$, for $\psi = \pi$ and $\psi = -\pi$, it yields the *outer circle* $(x + c)^2 + y^2 = (a + \mu)^2$ in the plane $z = 0$. The tube circle and the outer circle meet in the *pole* $p := (-\mu - c - a, 0, 0)$.

To get a rational parameterization of the cyclide without trigonometric functions, we use the identities

$$\cos\theta = \frac{1 - \tan^2\frac{\theta}{2}}{1 + \tan^2\frac{\theta}{2}} \quad \sin\theta = \frac{2\tan\frac{\theta}{2}}{1 + \tan^2\frac{\theta}{2}},$$

and set $u := \tan\frac{\phi}{2}, v := \tan\frac{\psi}{2}$. We write the obtained parametrization in homogeneous coordinates, i.e., the common denominator is written as a separate variable: Define $u_+ := 1 + u^2$, $u_- := 1 - u^2$, $v_+ := 1 + v^2$ and $v_- := 1 - v^2$ then $\hat{P} : \mathbb{R}^2 \to \mathbb{R}^4$ is given by

$$\begin{pmatrix} u \\ v \end{pmatrix} \mapsto \begin{pmatrix} \mu(cu_+v_+ - au_-v_-) + b^2u_-v_+ \\ 2u(av_+ - \mu v_-)b \\ 2v(cu_- - \mu u_+)b \\ au_+v_+ - cu_-v_- \end{pmatrix}$$

The image of $\hat{P}$ is the cyclide without the tube circle and the outer circle. Intuitively, the cyclide is cut along the outer circle and the tube circle, and "rolled out" to the plane. Therefore, we call the outer circle and the tube circle the *cut circles*. Paths on the cyclide crossing the cut circles correspond to paths in the parameter space crossing the infinite boundary. More precisely, an intersection with the tube (outer) circle causes a horizontally (vertically) asymptotic path in the parameter space. Paths passing through the cyclide's pole correspond to paths converging to one of the "corners" $(\pm\infty, \pm\infty)$ in parameter space.

3 Our implementation

We use the software framework implemented in CGAL's new `Arrangement_on_surface_2` package [3]. It provides an arrangement class that can be used to construct, maintain, overlay, and query two-dimensional arrangements on a parametric surface. It conceptually performs a sweep in the parameter space, i.e., a line $u = u_0$ is swept to the right through the parameter space.

Special diligence is needed for such curves at boundaries of the parameter space. The parameter space of the cyclide contains so called *identifications* of both pairs of opposite boundaries, i.e., for its parameterization P_S, it holds $\forall v \in V, P_S(u_{\min}, v) = P_S(u_{\max}, v)$ and $\forall u \in U, P_S(u, v_{\min}) = P_S(u, v_{\max})$, so for each point on the outer- and the tube-circle there exist two pre-images (four for the pole) in parameter space. This leads to problems for the sweep, since the event queue of the sweep line algorithms needs a unique order, and since only one DCEL-vertex should be constructed for each multiple pre-image. The modularity of CGAL's new `Arrangement_on_surface_2` package tackles these problems. To instantiate the package's main class, models of two concepts must be provided as template parameter.

First, the *GeometryTraits* fullfills the CGAL's ARRANGEMENTTRAITS_2 concept. It defines the types `Curve_2`, `X_monotone_curve_2`, and `Point_2`, and provides predicates and constructions on points and subcurves, e.g., lexicographic comparison of two points, or the construction of all intersection of two x-monotone curves.

Second, the *TopologyTraits* is responsible to determine the underlying DCEL-representation, to create the empty representation and to construct and maintain DCEL-features related to the boundary of the parameter space.

We describe next our models for both concepts:

GeometryTraits: We aim to represent the curves on the cyclide as algebraic curves in parameter space, and to realize the geometric predicates by computations in the parameter space.

For a cyclide with homogeneous parametrization

$\hat{P}$ and a surface F with homogeneous equation $\hat{F} \in \mathbb{Z}[x, y, z, w]$, the cut curve in the parameter space is implicitly defined by $f := \hat{F}(\hat{P}(u, v)) \in \mathbb{Z}[u, v]$. The resulting curve f has bidegree up to $(2 \cdot \deg F, 2 \cdot \deg F)$, hence we need a model of CGAL's ARRANGEMENTTRAITS_2 concept for algebraic curves in $\mathbb{R}^2$, regardless of their degree.

Such a model has recently been provided by Eigenwillig and Kerber [11] based on the observation that all required operations emerge from the topological and geometric analyses of single curves [12] and pairs of them. No condition is imposed on the input, i.e., curves can have arbitrary degree, and contain degeneracies, like covertical intersections, vertical asymptotes and isolated points.

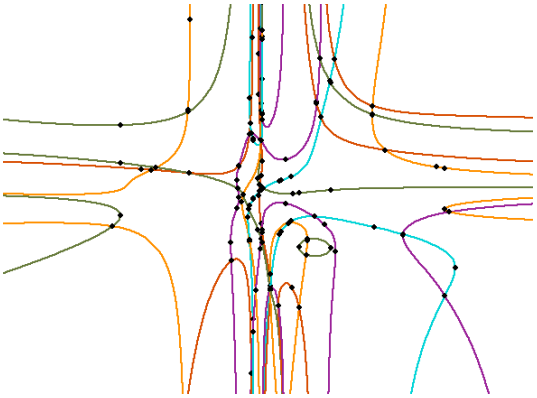


Figure 3.1: Cut-out of an arrangement in the parameter space of a cyclide, induced by 5 cubic surfaces

TopologyTraits: We provide a new TopologyTraits class for the cyclide. To handle the identifications at boundaries, it maintains two sorted sequences of DCEL-vertices to store the intersection of intersection curves with the cut circles. The position of such intersections is determined by horizontal and vertical asymptotes of curve-ends approaching infinity, the boundary of the parameter space.

Whenever the arrangement detects a curve-end approaching a cut circle, it asks the topology traits whether a DCEL-vertex is already stored for this position. If not, a new one is created and stored in the proper sequence; if yes, that one is used and the identification interactively takes place.

The analysis of curves contains the information about vertical asymptotes of curves (compare [12]), but not about horizontal asymptotes. We also implemented this step, with the following idea: there can be only finitely many positions where horizontal asymptotes might appear, since they are roots of a leading coefficient with respect to u. The curve-ends of arcs towards $u = \pm\infty$ can be assigned to such asymptotes by analysing the curve at some value u_0 "far" on the left or on the right.

Instance	#S	#V,#E,#F	t	t (2D)
ipl-1	10	119,190,71	**0.14**	0.14
ipl-1	20	384,682, 298	**0.58**	0.58
ipl-1	50	1837,3363,1526	**2.14**	2.00
ipl-2	10	358,575,217	**1.07**	1.25
ipl-2	20	1211,2147,937	**3.14**	3.04
ipl-3	10	542,847,305	**4.84**	4.62
ipl-3-6points	10	680,1092,412	**32.43**	31.17
ipl-3-2sing	10	694,1062,368	**5.82**	5.57
ipl-4	10	785,1204,419	**50.42**	49.97
ipl-4-6points	10	989,1529,540	**461.74**	450.54
ipl-4-2sing	10	933,1471,538	**53.01**	52.78

Table 1: Running times (in seconds) to construct arrangements on S_1 induced by algebraic surfaces

4 Results

We performed tests of our C++ implementation, executed on an AMD Dual-Core Opteron(tm) 8218 multi-processor Debian Etch platform, each core equipped with 1 MB internal cache and clocked at 1 GHz. The total memory consists of 32 GB. As compiler we used `g++` in version 4.1.2 with flags `-O2 -DNDEBUG`. Two results were computed for each instance, one that computes the arrangement using the *cyclidean topology* (`onSurface`), the other is computing the two-dimensional arrangement of the induced intersection curves *in parameter space*, i.e., with the topology of an unbounded plane (`Arrangements`). Our implementation allows to translate and rotate the reference cyclide in space. For the experiments presented in this work, we use the torus S_1 with $a = 2$, $b = 2$, $\mu = 1$, centered at the origin and the cyclide S_2 with $a = 13$, $b = 12$, $\mu = 11$, translated by a rational vector and rotated by a rational matrix.

We interpolated surfaces of fixed degree by randomly chosing points on a three-dimensional grid, having no or some degeneracies wrt S_1: the surfaces in "6points" instances share at least 6 common points, one of them is the pole of S_1. The surfaces in the "2sing" instances induce (at least) two singular intersections. The running times are listed in Table 1 that show good behavior of the implementation, even for higher degree surfaces. Degeneracies with respect to the reference surface result in higher running times as the instance "6points" shows. But this effect already appears in parameter space, as the last column indicates. In general, it is remarkable that in all tested instances, the spent time on the cyclides is (almost) identical to the computation of the curves in their parameter space. This let us conclude that the cyclidean topology is as efficient as the one for the unbounded plane and that the extra computation of horizontal asymptotes seems to be a cheap task. Most time is spent for geometric operations on algebraic curves. Thus, we infer that the chosen approach strongly hinges on the efficiency of the underlying 2D-implementation for arrangements of algebraic curves.

Instances	#S	#V,#E,#F	t
quadrics	10	428,646,219	**1.59**
degree-3	5	240,314,74	**1.56**
Overlay	-	942,1508,566	**1.91**
degree-3	10	794,1218,424	**6.25**
degree-4	10	325,418,93	**13.36**
Overlay	-	1623,2644,1021	**13.83**
degree-4	10	816,1188,372	**50.86**
degree-4	5	325,418,93	**13.52**
Overlay	-	1581,2488,907	**47.30**

Table 2: Running times (in seconds) to construct arrangements induced by algebraic surfaces of different degree on S_2, and to overlay them afterwards.

We also generated instances of random surfaces with degree up to 4 intersecting S_2, picked two of them, computed their arrangement and also the overlay of these arrangements. Reading Table 2, one sees that the overlay step is usually faster than the two initial constructions, as only a few new pairs of algebraic curves have to be analyzed newly.

Finally, we remark, that we also can immediately use other techniques implemented for CGAL's `Arrangement_on_surface_2`, such as point location, extending the DCEL by user data, and notifications.

5 Conclusion

Our work demonstrates the usefulness of generic programming: the combination of the planar arrangement algorithm for arbitrary curves with the software framework for arrangement on surfaces yields an arrangement algorithm for tori and Dupin cyclides almost immediately. New code was only written for the computation of the parameterized intersection curves, for the asymptotic behavior of infinite curve arcs, and for the topology traits of the cyclide. Relying on already tested and optimized code reduces the implementation effort, and makes the algorithm more robust and more efficient. We are already working on the adaptation of our traits classes with respect to the next version of the framework that will support geometric objects on identifications.

We also believe that the performance could be further improved: the computed arrangements often contain numerous vertically asymptotic arcs (compare Figure 3.1). The strategy proposed in [12] to shear non-regular curves and shearing back afterwards therefore results in a change of coordinates for many curves. A comparably efficient alternative approach that avoids to shear might be more suitable for this special subclass of curves.

Acknowledgements

We want to thank Ron Wein, who answered our questions on CGAL's arrangements in depth.

References

[1] L. Arge, M. Hoffmann, E. Welzl (eds.): *Algorithms - ESA 2007, 15th Annual European Symposium, Eilat, Israel, October 8-10, 2007, Proceedings*, *LNCS*, vol. 4698. Springer, 2007.

[2] J. L. Bentley, T. A. Ottmann: "Algorithms for Reporting and Counting Geometric Intersections". *IEEE Trans. on Computers* **C-28** (1979) 643–647.

[3] E. Berberich, E. Fogel, D. Halperin, K. Mehlhorn, R. Wein: "Sweeping and Maintaining Two-Dimensional Arrangements on Surfaces: A First Step". In: Arge et al. [1], 2007 645–656.

[4] E. Berberich, M. Hemmer, L. Kettner, E. Schömer, N. Wolpert: "An Exact, Complete and Efficient Implementation for Computing Planar Maps of Quadric Intersection Curves". In: *Proc. of the 21st Annual Symp. on Comput. Geom. (SCG 2005)*, 2005 99–106.

[5] E. Berberich, M. Kerber: "Exact Arrangements on Tori and Dupin Cyclides". In: *Proc. of the 2005 ACM Symp. on Solid and Physical Modeling, (SPM 08)*, 2008, to appear

[6] W. Boehm: "On Cyclides in Geometric Modeling". *Computer Aided Geometric Design* **7** (1990) 243–255.

[7] F. Cazals, S. Loriot: *Computing the exact arrangement of circles on a sphere, with applications in structural biology.* Technical Report 6049, INRIA Sophia-Antipolis, 2007.

[8] V. Chandru, D. Dutta, C. M. Hoffmann: "On the geometry of Dupin cyclides". *The Visual Computer* **5** (1989) 277–290.

[9] C. Dupin: *Applications de Géométrie et de Méchanique.* Bachelier, Paris, 1822.

[10] L. Dupont, M. Hemmer, S. Petitjean, E. Schömer: "Complete, Exact and Efficient Implementation for Computing the Adjacency Graph of an Arrangement of Quadrics". In: Arge et al. [1], 2007 633–644.

[11] A. Eigenwillig, M. Kerber: "Exact and Efficient 2D-Arrangements of Arbitrary Algebraic Curves". In: *Proc. of the Nineteenth Annual ACM-SIAM Symp. on Discrete Algorithms (SODA08)*, 2008 122–131.

[12] A. Eigenwillig, M. Kerber, N. Wolpert: "Fast and Exact Geometric Analysis of Real Algebraic Plane Curves". In: C. W. Brown (ed.) *Proocedings of the 2007 International Symposium on Symbolic and Algebraic Computation (ISSAC 2007)*, 2007 151–158.

[13] E. Fogel, D. Halperin, L. Kettner, M. Teillaud, R. Wein, N. Wolpert: "Arrangements". In: J.-D. Boissonnat, M. Teillaud (eds.) *Effective Computational Geometry for Curves and Surfaces*, chap. **1**, 1–66. Spinger, 2006.

[14] A. Forsyth: *Lectures on the Differential Geometry of Curves and Surfaces.* Cambridge Univ. Press, 1912.

[15] M. J. Pratt: "Cyclides in computer aided geometric design". *Computer Aided Geometric Design* **7** (1990) 221–242.

[16] M. J. Pratt: "Cyclides in computer aided geometric design II". *Computer Aided Geometric Design* **12** (1995) 131–152.

On the Topology of Planar Algebraic Curves

Jinsan Cheng* Sylvain Lazard* Luis Peñaranda*
Marc Pouget* Fabrice Rouillier† Elias Tsigaridas‡

Abstract

We introduce a method to compute the topology of planar algebraic curves. The curve may not be in generic position and may have vertical asymptotes. The algebraic tools are rational univariate representation for zero-dimentional ideals and multiplicities in these ideals. Experiments show the efficiency of our algorithm.

1 Introduction

We consider the problem of computing a geometric representation of a planar real algebraic curve $\mathcal{C}$, defined in a Cartesian coordinate system by a bivariate polynomial f with rational coefficients. More precisely, we address the problem of computing a planar graph whose vertices are mapped to points in the plane (possibly at infinity) and such that drawing the arcs as line segments gives a drawing isotopic to the input curve (see Figure 1). We assume that $\mathcal{C}$ is square-free and has no vertical lies. The assumption is a standard one in the context of this type of problems, while treating vertical lines is easy to deal with. There is no other assumptions on $\mathcal{C}$.

There have been many papers addressing the problem of computing the topology of algebraic plane curves [2, 3, 4, 8]. All algorithms use a sheared curve if it is not in generic position.

All these algorithms perform the following phases. (1) Project the x-critical points of the curve on the x-axis, using resultants or Sturm-Habicht sequences, and isolate the real roots of the resulting univariate polynomial in x. This gives the x-coordinates of all the x-critical points. (2) For each such value x_i, compute the intersection points between the curve $\mathcal{C}$ and the vertical line $x = x_i$. (3) Through each of these points, determine the number of branches of $\mathcal{C}$ coming from the left and going to the right. (4) Connect all these points appropriately.

The main difficulty in all these algorithms is to compute efficiently all the critical points in Phase 2 because the x-critical values in Phase 1 are, a priori, non-rational thus computing the corresponding y-coordinates in Phase 2 amounts, in general, to solving a univariate polynomial with non-rational coefficients and at least a multiple root (corresponding to the critical point).

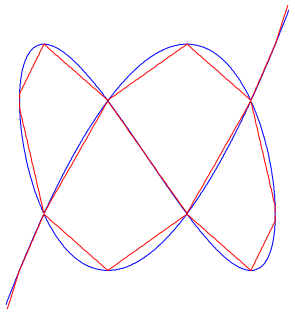


Figure 1: $\mathcal{C} : 16\,x^5 - 20\,x^3 + 5\,x - 4\,y^3 + 3\,y = 0$ plotted in MAPLE and its isotopic graph computed by ISOTOP.

Our contributions. Unlike most previous algorithms, this algorithm handles curves in non-generic positions in the given Cartesian coordinate system without shearing. The other originality of our approach is that we succeed to avoid the computation of sub-resultant sequences. As a result, our MAPLE implementation, ISOTOP, substantially outperforms previous available MAPLE implementations, TOP [3] and INSULATE [8]. Recall that TOP requires an initial precition to be set, thus the results may not correct if the precision is not set appropriately. INSULATE is certified but the critical points may be computed in a sheared coordinate system. We observe on preliminary tests that the running time of our implementation is remarkably stable. Furthermore, we observe, on 24 curves of degree between 4 and 13, that ISOTOP is, on average, 8.34 times faster than INSULATE with a median of 4.3 and extrema of 0.8 and 43.9. Compared to TOP using 60 (resp. 500) digits of initial precision, ISOTOP is, on average, 10.36 (resp. 26.7) times faster with a median of 1.2 (resp. 4.25) and extrema of 0.2 and 103.3 (resp. 0.7 and 193.2).

The novelty of our algorithm mainly relies upon the use of three new ingredients for this problem. First, we use a formula of Teissier relating the multiplicities of the roots of a polynomial or a system, which avoids

*LORIA (INRIA, CNRS, Nancy Université) and INRIA Nancy-Grand Est, Nancy, France. Firstname.Name@loria.fr
†LIP6 (Université Paris 6, CNRS) and INRIA Paris-Rocquencourt, Paris, France. Fabrice.Rouillier@lip6.fr
‡INRIA Sophia-Antipolis Méditerranée. Elias.Tsigaridas@inria.fr

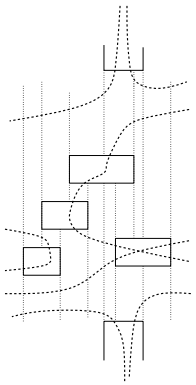


Figure 2: Example of rectangle decomposition of the plane induced by the isolating boxes (critical and asymptotes).

computing Sturm-Habicht type sequences.

Second, we isolate the roots of bivariate systems using (i) Gröbner bases, (ii) Rational Univariate Representations (RUR) , and (iii) a subdivision technique based on Descartes' rule for isolating the roots of the univariate polynomials.

Third, we present a variant of the standard combinatorial part of the algorithm for computing the topology. This variant avoids computing points of the curve on (rational) vertical lines between the critical points. Alternatively, we compute a decomposition of the plane by rectangular boxes containing at most one critical point. To achieve the connection in each box, we use their multiplicities in the system of critical points.

We also present an output-sensitive bound on the worst-case complexity of our algorithm. To the best of our knowledge, this is, for the problem considered here, the first time that the complexity of a (certified) algorithm based on refinements and approximations is analyzed. Our technique, even though not novel, could presumably also be used to analyze the complexity of some previous algorithms.

2 Notations and preliminaries.

Let $\mathcal{C}_f$ be a real algebraic plane curve defined by a bivariate polynomial f in $\mathbb{Q}[x, y]$.

Let f_x denote the derivative of f with respect to x and f_{y^k} (sometimes also f_k) denote the k^{th} derivative with respect to y. A point $\mathbf{p} = (\alpha, \beta) \in \mathbb{C}^2$ is called x-critical (simply called critical) if $f(\mathbf{p}) = f_y(\mathbf{p}) = 0$, singular if $f(\mathbf{p}) = f_x(\mathbf{p}) = f_y(\mathbf{p}) = 0$, and x-extreme (simply called extreme) if $f(\mathbf{p}) = f_y(\mathbf{p}) = 0$ and $f_x(\mathbf{p}) \neq 0$. Similarly are defined y-critical and y-extreme points.

The ideal generated by polynomials $P_1, \ldots, P_i$ is denoted $\mathbb{I}(P_1, \ldots P_i)$. In the following, we often identify the ideal and the system of equations $\{P_1 = 0, \ldots, P_i = 0\}$ (or any equivalent system induced by a set of generators of the ideal). Let $I_c = \mathbb{I}(f, f_y)$ and $I_s = \mathbb{I}(f, f_x, f_y)$. Their roots are, respectively, the x-critical and singular points of $\mathcal{C}$. We also consider the Jacobian ideal $I_j = \mathbb{I}(f_x, f_y)$ and its associated ideal $I_m = \mathbb{I}(f_x/\gcd(f_x, f_y), f_y/\gcd(f_x, f_y))$ called the Milnor ideal.

We now recall the notion of multiplicity of the roots of an ideal, then we state two lemmas using this notion for studying the local topology at singular points. Geometrically, the notion of multiplicity of intersection of two regular curves is intuitive. If the intersection is transverse, the multiplicity is one; otherwise, it is greater than one and it measures the level of degeneracy of the tangential contact between the curves. Defining the multiplicity of the intersection of two curves at a point that is singular for one of them (or possibly both) is more involved and an abstract and general concept of multiplicity in an ideal is needed.

Definition 1 ([1]) *Let I be an ideal of $\mathbb{Q}[x, y]$ and $\overline{\mathbb{Q}}$ the algebraic closure of $\mathbb{Q}$. To each zero (α, β) of I corresponds a local ring $(\overline{\mathbb{Q}}[x, y]/I)_{(\alpha,\beta)}$ obtained by localizing the ring $\overline{\mathbb{Q}}[x, y]/I$ at the maximal ideal $\mathbb{I}(x-\alpha, y-\beta)$. When this local ring is finite dimensional as $\overline{\mathbb{Q}}$-vector space, we say that (α, β) is an isolated zero of I and this dimension is called the* **multiplicity of (α, β) as a zero of I**.

Let $f, g \in \mathbb{Q}[x, y]$ be such that the intersection of $\mathcal{C}_f$ and $\mathcal{C}_g$ in $\mathbb{C}^2$ contains a zero-dimensional component equal to point $\mathbf{p} = (\alpha, \beta)$. Then (α, β) is an isolated zero of $\mathbb{I}(f, g)$ and its multiplicity, denoted by $Int(f, g, \mathbf{p})$, is called the **intersection multiplicity of the two curves at this point**. *A singular point of a curve $\mathcal{C}_f$ is an isolated zero of the Jacobian ideal $I_j = \mathbb{I}(f_x, f_y)$ and the multiplicity of this point as a zero of this system is called the* **Milnor number** *of the singular point.*

We call a **fiber** *a vertical line of equation $x = \alpha$. For a point $\mathbf{p} = (\alpha, \beta)$ on the curve $\mathcal{C}_f$, we call the multiplicity of β in the univariate polynomial $f(\alpha, y)$ the* **multiplicity of p in its fiber** *and denote it as $mult(f(\alpha, y), \beta)$.*

Lemma 1 ([7]) *For a singular point $\mathbf{p} = (\alpha, \beta)$ of the curve $\mathcal{C}_f$, we have*

$$mult(f(\alpha, y), \beta) = Int(f, f_y, \mathbf{p}) - Int(f_x, f_y, \mathbf{p}) + 1. \quad (1)$$

For an x-extreme point, the same formula holds and, as the Milnor number vanishes, it simplifies into

$$mult(f(\alpha, y), \beta) = Int(f, f_y, \mathbf{p}) + 1. \quad (2)$$

Lemma 2 ([8]) *Let $\mathbf{p} = (\alpha, \beta)$ be a real singular point of the curve $\mathcal{C}_f$ of multiplicity k in its fiber. Let*

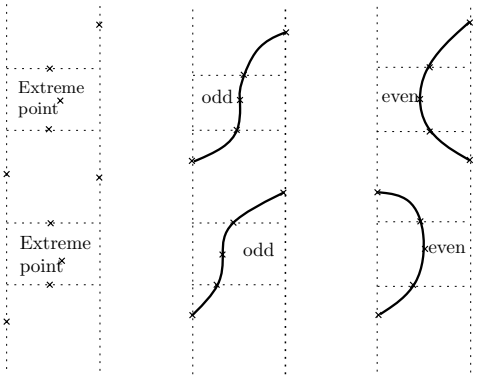


Figure 3: Different connections according to multiplicities for the same crossing pattern.

B be a box satisfying (i) B contains **p** *and no other x-critical point, (ii) the function f_{y^k} does not vanish on B, and (iii) the curve $\mathcal{C}_f$ crosses the border of B only on the left or the right sides. Then the topology of the curve in B is given by connecting the singular point with all the intersections on the border.*

Given a zero-dimensional ideal $I = \mathbb{I}(P_1, \ldots, P_s)$ where the $P_i \in \mathbb{Q}[x_1, \ldots, x_n]$, a Rational Univariate Representation (RUR) [6] of the solutions $\mathrm{V}(I)$ is given by $F(t) = 0, x_1 = \frac{G_1(t)}{G_0(t)}, \ldots, x_n = \frac{G_n(t)}{G_0(t)}$, where $F, G_0, \ldots, G_n$ are univariate polynomials in $\mathbb{Q}[t]$ (where t is a new variable). All these univariate polynomials, and thus the RUR, are uniquely defined with respect to a given polynomial $\gamma \in \mathbb{Q}[x_1, \ldots, x_n]$ which is injective on $V(I)$; γ is called the **separating polynomial** of the RUR. [1]

3 Algorithm

Input. A square free bivariate polynomial $f \in \mathbb{Q}[x, y]$ without factor in $\mathbb{Q}[x]$.

Step 1. *Isolating boxes of the singular points and of the x-extreme points.* First compute the RURs of I_c and I_m with the same separating polynomial γ. Let F_c and F_m denote the univariate polynomials of the RURs of I_c and I_m. Isolate the roots of F_c and those of the gcd of F_c and F_m and match the resulting intervals. The roots of F_c that are roots of $\gcd(F_c, F_m)$ map to the singular points of $\mathcal{C}$ and the other roots of F_c map to the x-extreme points of $\mathcal{C}$. Using the RUR of I_c, compute isolating boxes of the singular points and of the x-extreme points. Note that these boxes are pairwise disjoint since they isolate the x-critical points of $\mathcal{C}$ (see Figure 2).

[1] The polynomial F is the characteristic polynomial of m_γ, the multiplication operator by the polynomial γ, in $\mathbb{Q}[x_1, \ldots, x_n]/I$.

Step 2. *Refinement of the isolating boxes of the x-extreme points.* For each vertical or horizontal side of each such box, B, isolate its intersections with $\mathcal{C}$ and refine the box (by refining the corresponding isolating interval of F_c) until there are two intersection points on the border of B. We further refine until there is at most one crossing on the top (resp. bottom) side of B. Unlike comparable algorithms, we do not require that $\mathcal{C}$ intersects the boundary of B only on its vertical sides.

Step 3. *Refinement of the isolating boxes of the singular points.* We refine these boxes exactly as in [8] (see Lemma 2) except for the way the multiplicity k of each singular point in its fiber is computed. We deduce k from the Teissier formula (see Lemma 1) and from the RURs computed above.

Consider each singular point, **p**, in turn. Compute the multiplicities k_c and k_m of the root associated to **p** in the univariate polynomials F_c and F_m, respectively. This gives the multiplicity of **p** in I_c and I_m because the RURs preserve multiplicities. By the Teissier formula, the multiplicity of **p** in its fiber is $k = k_c - k_m + 1$. Refine the box containing **p** until f_{y^k} does not vanish in the box. Further refine the x-coordinates of the box until $\mathcal{C}$ only intersects the vertical boundary of the box.

Step 4. *Vertical asymptotes.* If there exist vertical asymptotes, we can work as follows.

First, compute an upper bound M_y on the absolute value of the y-coordinates of the y-critical points (this is of course done without computing these critical points). Compute also a bound M_x on the absolute value of the y-coordinates of the x-critical points (which are already computed). Isolate the roots of the polynomial in x obtained as the leading coefficient of f seen as a polynomial in y. For each root α we have an isolating interval $[a, b]$. Substitute $x = a$ (resp. $x = b$) in f and deduce an upper bound, M_α, on the absolute value of the y-coordinates of the intersection of $\mathcal{C}$ and $x = a$ (resp. $x = b$). Set $M = \max(M_\alpha, M_x, M_y)$. Then, a branch crossing the segment $]a, b[\times M$ (resp. $]a, b[\times -M$) goes to $+\infty$ (resp. $-\infty$) with asymptote $x = \alpha$. Finally, we determine whether a given branch is to the left or to the right of the asymptote by comparing the x-coordinates of the asymptote and the crossing point (see Figure 2).

Step 5. *Connections.* For simplicity, all the boxes computed above are called critical boxes and the points at infinity on vertical asymptotes are also called critical. First compute, with a sweep-line algorithm, the vertical rectangular decomposition obtained by extending the vertical sides of the critical boxes either to infinity or to the first encountered critical box (see Figure 2). On each of the edges of the decomposition, isolate the intersections with $\mathcal{C}$. Create vertices in the graph corresponding to these intersection points and

to the critical points. For describing the arcs connecting these vertices in the graph, we assimilate, for simplicity, the points and the graph vertices. In each critical box, connect the critical point to the points on the boundary of the box.

For computing the connections in the non-critical rectangles of the decomposition, we use the multiplicities of the extreme points and a greedy algorithm. According to Lemma 1, the multiplicity k of an extreme point $\mathbf{p}$ in its fiber is one plus the multiplicity of $\mathbf{p}$ in I_c. Recall that the multiplicity of $\mathbf{p}$ in I_c is the multiplicity of the corresponding root in the polynomial F_c of the RUR. All the multiplicities of the extreme points in their fibers can thus be efficiently computed.

The geometric meaning of the parity of this multiplicity is the following: if it is even, the curve makes a U-turn at the extreme point; else, the curve is x-monotone in the neighborhood of the extreme point. Still, there are some difficulties for connecting the vertices, as illustrated on Figure 3: on the left is the information we may have on the crossings for two extreme points with x-overlapping boxes; the second and third drawings are two possible connections *in the middle rectangle* for given parities of the multiplicities. To distinguish between these two situations we compute the connections in rectangles starting from the top such that the connections in a rectangle below a critical box are computed once the connections in all the rectangles above the box are done.

First, the connections in the unbounded rectangles above critical boxes are straightforward: the connections between the vertices on the two vertical walls are one-to-one starting from infinity and if a vertex remains on a vertical wall, there is a vertex on the horizontal wall which it has to be connected with. Now, once all the connections have been computed in the rectangle(s) above the box of an extreme point, these connections and the multiplicity of the extreme point yield the connections in the rectangle(s) below. Indeed, if there is a vertex on the bottom side of the critical box, it lies on the top side of a rectangle and, inside this rectangle, the vertex is connected to the topmost vertex on the left or right wall, depending on the multiplicity of the extreme point and on the side of the connection of the branch above the extreme point; the other connections in this rectangle and in the possible other rectangles below the critical box are performed similarly as for unbounded rectangles.

Note that the two unbounded rectangles that are vertical half-planes are treated separately in a straightforward manner: for each vertex on the vertical wall is associated an arc going to infinity.

Output. A graph isotopic to the curve is output. In addition x-critical, singular points and vertical asymptotes are identified and their position is approximated by boxes.

4 Complexity analysis

Consider a real algebraic plane curve defined by a square-free polynomial $f \in \mathbb{Z}[x, y]$ of degree bounded by d and maximum coefficient bit-size bounded by τ. Let s_k be the bit-size of the separation bound of f and all its derivatives with respect to y, s_c be the bit-size of the separation bound between all the critical points of f, $s = \max\{s_c, s_k\}$ and R be the number of critical points. Let $\widetilde{\mathcal{O}}_B$ denote the bit complexity where the polylogarithmic factors are omitted. For reasons of space we omit the proof of the following theorem.

Theorem 3 *Our algorithm computes the topology in* $\widetilde{\mathcal{O}}_B\left(R(d^{12}\tau s + d^{12}s^2 + d^7\tau^2)\right)$ *time, which is in* $\widetilde{\mathcal{O}}_B(N^{22})$, *where* $N = \max\{d, \tau\}$.

Acknowledgments

We would like to thank N. Wolpert and L. González-Vega for kindly supplying their code.

References

[1] D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry*. Number 185 in GTM. Springer, New York, 2nd edition, 2005.

[2] A. Eigenwillig, M. Kerber, and N. Wolpert. Fast and exact geometric analysis of real algebraic plane curves. In C. W. Brown, editor, *Proc. Int. Symp. Symbolic and Algebraic Computation*, pages 151–158, Waterloo, Canada, 2007. ACM.

[3] L. González-Vega and I. Necula. Efficient topology determination of implicitly defined algebraic plane curves. *Computer Aided Geometric Design*, 19(9), 2002.

[4] H. Hong. An efficient method for analyzing the topology of plane real algebraic curves. *Mathematics and Computers in Simulation*, 42(4–6):571–582, 1996.

[5] B. Mourrain, S. Pion, S. Schmitt, J.-P. Técourt, E. P. Tsigaridas, and N. Wolpert. Algebraic issues in Computational Geometry. In J.-D. Boissonnat and M. Teillaud, editors, *Effective Computational Geometry for Curves and Surfaces*, Mathematics and Visualization, Chapter 3. Springer, 2006.

[6] F. Rouillier. Solving zero-dimensional systems through the rational univariate representation. *Journal of Applicable Algebra in Engineering, Communication and Computing*, 9(5):433–461, 1999.

[7] B. Teissier. Cycles évanescents, sections planes et conditions de whitney. (french). In *Singularités á Cargèse (Rencontre Singularités Géom. Anal., Inst. Études Sci., Cargèse, 1972)*, number 7–8 in Asterisque, pages 285–362. Soc. Math. France, Paris, 1973.

[8] N. Wolpert and R. Seidel. On the Exact Computation of the Topology of Real Algebraic Curves. In *Symposium of Computational Geometry*. ACM, 2005.

Topological Considerations for the Incremental Computation of Voronoi Diagrams of Circular Arcs

Martin Held* Stefan Huber†

Abstract

We study the computation of Voronoi diagrams of points, straight-line segments and circular arcs in the two-dimensional Euclidean plane. Our algorithm is based on a randomized incremental insertion of the sites and makes use of the topology-oriented approach by Sugihara et alii. It was implemented and integrated into the Voronoi package VRONI. However, in this extended abstract we focus only on the topological and graph-theoretic details of an insertion.

1 Introduction

1.1 Motivation

Voronoi diagrams of straight-line segments turned out to be useful in a variety of applications with a geometric flavor. We note that supporting circular arcs is important for the practical application of a Voronoi algorithm: offsetting a polygon introduces circular arcs, and it is generally requested that the result of an offsetting operation can again be used as input for a Voronoi algorithm. Handling circular arcs as genuine arcs is imperative in the PCB[1] business, since PCB data may be huge; typically, one cannot afford to replace every arc by tens or even hundreds of straight-line segments as this would cause the memory footprint of a Voronoi-based application to sky-rocket.

1.2 Basic definitions

For two points $p, q \in \mathbb{R}^2$, let $d(p,q)$ denote the Euclidean distance between p and q. If $Q \subset \mathbb{R}^2$ is a set then $d(p,Q)$ is defined as $\inf\{d(p,q) : q \in Q\}$. Similarly, if $\mathcal{Q}$ is a set of a finite number of sets, then $d(p,\mathcal{Q}) := \min_{Q\in\mathcal{Q}} d(p,Q)$.

In the sequel we explain how to compute $\mathcal{VD}(S)$, where S is a set of points, straight-line segments and circular arcs. For technical reasons, we regard a line segment or a circular arc as the union of three objects: an open segment/arc and its two end points. Furthermore, we assume that every arc is oriented counterclockwise (CCW), and that no arc is greater than a semi-circle[2]. Points, open straight-line segments and open circular arcs are called *sites*. If for every open segment and arc of S its end points also belong to S and if no sites intersect pairwise then S is called a *proper* input set.

For a vector v and a point p, let $H(p,v)$ be the half-plane $\{q \in \mathbb{R}^2 : q \cdot v \geq p \cdot v\}$. The result of the rotation of v around the origin by 90^o is denoted by v^{CCW}, while v^{CW} stands for a rotation by -90^o. Following [3], the *cone of influence* $\mathcal{CI}(s)$ of a site s is defined as $\mathcal{CI}(s) := \mathbb{R}^2$ if s is a point, $\mathcal{CI}(s) := H(a, b-a) \cap H(b, a-b)$ if s is a segment with end points a and b, and $\mathcal{CI}(s) := H(c, (s-c)^{\mathrm{CCW}}) \cap H(c, (e-c)^{\mathrm{CW}})$ if s is an arc centered at c with start point s and end point e. We define the *Voronoi cell* of a site $s \in S$ as $\mathcal{VC}(s,S) := \mathrm{cl}\{q \in \mathrm{int}\,\mathcal{CI}(s) : d(q,s) \leq d(q,S)\}$, where $\mathrm{int}\,Q$ denotes the (topological) interior of the set Q and $\mathrm{cl}\,Q$ stands for the closure of Q. (The consideration of the interior and exterior in the definition of $\mathcal{VC}(s,S)$ is a technical twist[3] in order to avoid undesired "one-dimensional" portions of a Voronoi cell if two circular arcs meet tangentially in an end point such that the interiors of their cones of influence overlap.) The *Voronoi polygon* $\mathcal{VP}(s,S)$ is given by the boundary of $\mathcal{VC}(s,S)$, and the *Voronoi diagram* $\mathcal{VD}(S)$ of S is defined (as usual) as $\mathcal{VD}(S) := \bigcup_{s\in S} \mathcal{VP}(s,S)$. For two sites $s_1, s_2 \in S$, the *bisector* $b(s_1,s_2)$ is defined as the loci of points out of $\mathcal{CI}(s_1) \cap \mathcal{CI}(s_2)$ which are equidistant to s_1 and s_2. A *Voronoi edge* between s_1, s_2 is a connected portion of $\mathcal{VP}(s_1,S) \cap \mathcal{VP}(s_2,S)$; it lies on $b(s_1,s_2)$. *Voronoi nodes* are points where three or more Voronoi edges meet. The *clearance disk* $\mathcal{CD}(p,S)$ of a point $p \in \mathbb{R}^2$ is the closed disk centered at p with *clearance* radius $r := d(p,S)$.

1.3 Prior and related work

A worst-case optimal $O(n \log n)$ algorithm for the computation of the Voronoi diagram of n points, straight-line segments and circular arcs was intro-

*Universität Salzburg, FB Computerwissenschaften, A–5020 Salzburg, Austria; held@cosy.sbg.ac.at; Partially supported by Austrian FWF Grants L43-N12 and L367-N15.

†Universität Salzburg, FB Computerwissenschaften, A–5020 Salzburg, Austria; shuber@cosy.sbg.ac.at; Supported by Austrian FWF Grant L43-N12.

[1] Printed-circuit board.

[2] We split arcs greater than semi-circles.

[3] Yap [5] resorts to ϵ-neighborhoods. Alt and Schwarzkopf [1] consider cells that are partially open; as a consequence, the intersection between adjacent cells may be empty.

duced by Yap [5]. At least in theory, Fortune's sweepline algorithm [2] is also applicable to circular arcs. However, for both algorithms we are not aware of an actual implementation that handles circular arcs.

More recently, Alt and Schwarzkopf [1] studied Voronoi diagrams of so-called "harmless sites" (which include circular arcs). They first select one point out of the relative interior of each curve and then insert the curves in a randomized order, obtaining an expected running time of $O(n \log n)$. However, their paper focuses mostly on establishing a theoretical basis for the definition of Voronoi diagrams of planar curves, while the actual algorithmic aspect of the insertion of a curve is only sketched. In any case, no implementation of their algorithm is known.

1.4 Survey of the Voronoi algorithm

The success of Held's Voronoi package VRONI [3] motivated us to extend its construction scheme to points, straight-line segments and circular arcs in the two-dimensional Euclidean plane. Once again we resort to the topology-oriented approach by Sugihara et al. [4]. Starting with an initially empty set of processed sites, the final Voronoi diagram is obtained by incrementally adding one new site at a time to the set of processed sites and updating the Voronoi diagram accordingly.

Every update of the Voronoi diagram is performed by deleting old Voronoi nodes (and creating new Voronoi nodes). As in the case of segment Voronoi diagrams, care has to be taken in order to prevent the removal of cycles of Voronoi edges while deleting Voronoi nodes during an incremental update. However, the insertion of circular arcs causes problems to surface that do not occur for segment Voronoi diagrams. In the sequel, we discuss all topological and graph-theoretical extensions of the incremental insertion needed for handling circular arcs.

Our new algorithm has been implemented in ANSI C and integrated into VRONI. We emphasize, though, that the basic scheme presented below for incrementally inserting a circular arc into a Voronoi diagram is not bound to the limits of VRONI.

2 The algorithm

Let S be a set of sites and consider an open arc $s \notin S$ that is to be inserted into $\mathcal{VD}(S)$. Let $S^+ := S \cup \{s\}$. We assume that S^+ is a proper input set and that $\mathcal{VD}(S)$ is known. In order to insert the arc s into $\mathcal{VD}(S)$, we proceed as follows:

1. We mark a Voronoi node ("seed node") of $\mathcal{VD}(S)$ whose clearance disk is intersected by s.

2. We scan $\mathcal{VD}(S)$ and mark all other nodes whose clearance disks are intersected by s.

3. We remove all those Voronoi edges of $\mathcal{VD}(S)$ which have both nodes marked, and compute new nodes[4] on the edges with only one node marked.

While the second task boils down to a simple graph traversal that is identical to the case of segment Voronoi diagrams [3], the first and the third task require some caution, as explained below.

2.1 Selecting a seed node

A *seed node* is a node of $\mathcal{VD}(S)$ which lies in $\mathcal{VC}(s, S^+)$. Thus, its clearance disk is intersected by s and it needs to be removed. We search for a seed node in $\mathcal{VP}(p, S) \cap \mathcal{CI}(s)$, where p is the start point of s. If one or more candidates for a seed node exist in $\mathcal{VP}(p, S) \cap \operatorname{int} \mathcal{CI}(s)$, then we select the node whose clearance disk is violated the most: We pick the node v such that $d(v, S) - d(v, s)$ is minimized. A second seed node is selected within the Voronoi polygon of the end point of s. In any case, we do not select a node as seed node, or mark it in the subsequent scan of $\mathcal{VD}(S)$, if it coincides with the start or end point of s. If, however, no node of $\mathcal{VP}(p, S)$ lies within $\operatorname{int} \mathcal{CI}(s)$ then one can prove that there exist nodes of $\mathcal{VP}(p, S)$ on $\operatorname{bd} \mathcal{CI}(s)$, and we distinguish the following cases.

2.1.1 Selecting a seed node in the presence of tangential sites

Suppose that s meets exactly one site $s' \in S$ tangentially in the common end point p. Let e_1, e_2 be the two Voronoi edges that emanate from p, see Fig. 1. We note that e_1 and e_2 lie on the same supporting line g through p.

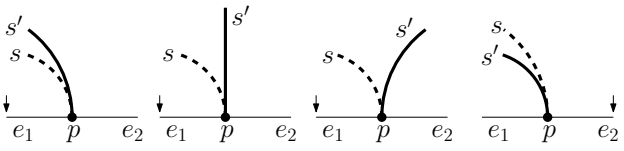


Figure 1: Selecting a seed node if sites meet tangentially.

The start node shared by e_1 and e_2 is excluded from further consideration because we do not select as seed node a node that coincides with the point p. Since $\mathcal{VP}(p, S) \cap \mathcal{CI}(s) \subset g$, the two other nodes on e_1 and e_2 are the only nodes of $\mathcal{VP}(p, S)$ that can be selected as seed node. Suppose that the center of s is on the side of e_1 relative to g and p. We base our decision on the relative order of the sites incident upon p:

- Case: s' is an arc.
  If the center of s' is on the side of e_2 then the node on e_1 is admissible as seed node. If the

[4] The computation of the new Voronoi nodes is explained in the full version of this paper.

center of s' is on the side of e_1 and the radius of s' is greater than the radius of s, then the node on e_1 is admissible; otherwise, the node on e_2 is admissible.

- Case: s' is a segment.
  The node on e_1 is admissible.

In Fig. 1 the little arrows point to the edge on which we select the seed node. If, however, more sites are incident upon p then we need to handle spikes; see below.

2.1.2 Selecting a seed node in the presence of spikes

Suppose that several segments or arcs meet in a common end point p. We scan all nodes of the Voronoi polygon $\mathcal{VP}(p, S)$. If $\mathcal{VP}(p, S)$ has a node v with a clearance greater than zero then v does not coincide with p. In this case we proceed as normal: if v lies on $\mathrm{bd}\,\mathcal{CI}(s)$ then we evaluate $d(v, S) - d(v, s)$. If some sites meet tangentially at p we also have to check whether v is admissible; see above. Again, we select that (admissible) node as seed node whose clearance disk is violated the most.

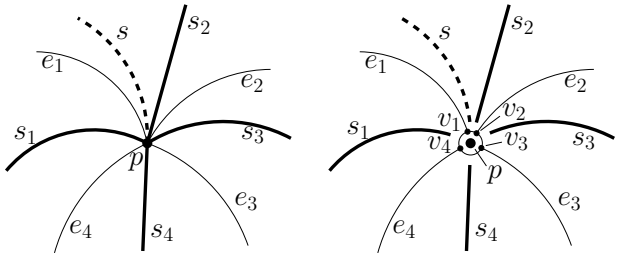


Figure 2: Selecting a seed node when multiple sites meet in a common end point. Left: Geometric view. Right: Topological view.

Otherwise, if no such (admissible) node exists, then we scan the Voronoi edges that are incident upon the nodes which coincide with p. (In Fig. 2, these nodes are numbered $v_1, \ldots, v_4$.) For such a node v_i we consider the Voronoi edge e_i incident upon v_i whose second node does not belong to $\mathcal{VP}(p, S)$. (If no such edge is incident upon v_i then we originate a recursive search in $\mathcal{VD}(S)$, starting at v_i.) For every such edge e_i it is tested whether s intersects the clearance disk of its second node. (One can prove[5] that such a suitable node always exists.)

We emphasize that nodes which coincide with an input point are never deleted during an incremental update. Therefore, it is guaranteed that in the final Voronoi diagram every point site will have a Voronoi region associated with it; it may have zero area, though.

[5] All proofs are given in the full version of this paper.

2.2 Removing a tree of Voronoi edges

Assume that two seed nodes have been determined. Starting at one seed node we recursively scan the Voronoi diagram $\mathcal{VD}(S)$ and mark all those nodes whose clearance disks are intersected by s. Obviously, all those nodes need to be deleted since they cannot belong to $\mathcal{VD}(S^+)$. Similarly, it seems natural to remove a Voronoi edge if both of its nodes have been marked for deletion. However, we have to check whether a cycle exists within the portion T of $\mathcal{VD}(S)$ which is marked for deletion. It can be shown that those portions of edges of $\mathcal{VD}(S)$ which are completely contained in $\mathcal{VC}(s, S^+)$ form a tree. In other words, T contains a cycle if and only if T contains an edge of $\mathcal{VD}(S)$ which has both nodes marked but needs to be preserved partly. Figure 3 depicts a (dashed) circular arc whose insertion would cause the removal of all nodes of the Voronoi cells of its two end points.

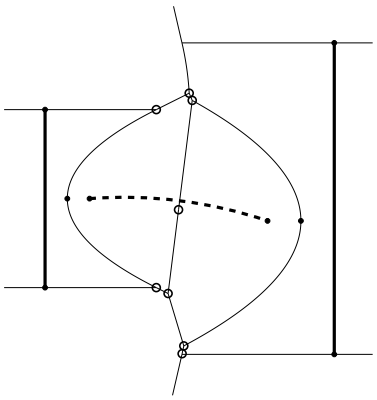


Figure 3: The insertion of the dashed arc causes the nodes depicted by circles to be marked. Splitting the two parabolic arcs at their apices avoids the (incorrect) removal of two Voronoi cells.

Suppose that e is an edge whose two nodes are marked for deletion but which needs to be preserved partly. We can distinguish two cases, depending upon whether or not e is completely contained in $\mathcal{CI}(s)$.

2.2.1 Voronoi edge partly outside of cone of influence

Consider a Voronoi edge and assume that the apex of its supporting conic lies on the edge. As suggested in [3], we insert a degree-two node in order to split a conic Voronoi edge at its apex. (See Fig. 3.) Hence, for the sequel we may assume that no Voronoi edge has the apex of its supporting conic in its (relative) interior. Then one can prove that every Voronoi edge of a segment Voronoi diagram either does not have both nodes marked for deletion or is completely contained in the current cone of influence of the new segment to be inserted. Unfortunately, once we deal with circular arcs the insertion of apex nodes is of limited

help for avoiding cycles: even if all Voronoi edges are split at the apices of their supporting conics, it still is possible that both nodes of an edge e are marked for deletion while e is not completely contained in $CI(s)$, as illustrated in Fig. 4.

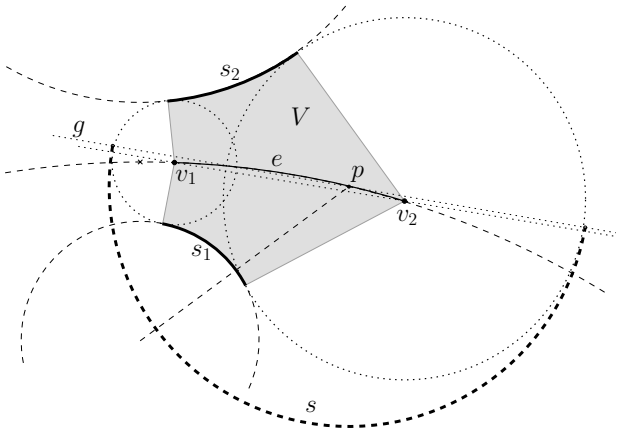


Figure 4: The insertion of the arc s causes both nodes of e to be marked although some portion of e lies outside of $CI(s)$.

This sample arrangement of input sites is constructed as follows: We consider the hyperbolic bisector between two circles that are disjoint. We choose two arcs s_1, s_2 on these circles and the corresponding Voronoi edge e on their bisector such that e does not contain the apex in its interior and such that $\mathcal{VC}(s_1, S)$ contains all secants of e. We denote by v_1 (resp. v_2) that node of e which has smaller (resp. larger) clearance. We want to insert an arc s such that the clearance disks of v_1 and v_2 are intersected by s, even though e is not completely contained in $\mathcal{CI}(s)$.

Let V be the union of all line segments resulting from the normal projection of points of e onto s_1 and s_2. Now consider the supporting line g of a secant of e that is parallel to the line through v_1 and v_2: We get that $g' := g \setminus \text{int}(V \cup \mathcal{CD}(v_1) \cup \mathcal{CD}(v_2))$ consists of two parts because the set $g \cap \text{int}(V \cup \mathcal{CD}(v_1) \cup \mathcal{CD}(v_2))$ is connected. Within each part of g' we choose a point close enough to the neighboring clearance disk such that the line through this point orthogonal to g intersects this clearance disk. All that remains to do is to use these two points as the end points of a semi-circle (which has to lie on that side of g which contains s_1). By construction, s intersects the clearance disks of v_1 and v_2. Also by construction, some portion of e does not lie on the side of v_1, v_2 relative to g. Thus, this portion of e is outside of $CI(s)$! This construction scheme can be adapted to nearly every combination of input sites s_1, s_2 as long as e does not take on the form of a straight-line segment.

We solve this problem by inserting a dummy degree-two node p that breaks up the cycle: We always find a proper point p on e by considering the normal projection of the center of s onto s_1, and by intersecting the resulting projection line with e. The resulting node p need not lie outside of $\mathcal{CI}(s)$ but one can prove that it will never lie in the future Voronoi cell $\mathcal{VC}(s, S^+)$.

2.2.2 Voronoi edge completely contained in cone of influence

Now suppose that e is a Voronoi edge of $\mathcal{VD}(S)$ which does not lie completely within $\mathcal{VC}(s, S^+)$ although both of its nodes are marked and although it is contained completely in $\mathcal{CI}(s)$. As Fig. 5 illustrates, the site s_1 enclosed by the cycle that contains the Voronoi edge e may be a segment or arc. Both nodes of e are marked even though a point p exists on e whose clearance disk is not violated by s.

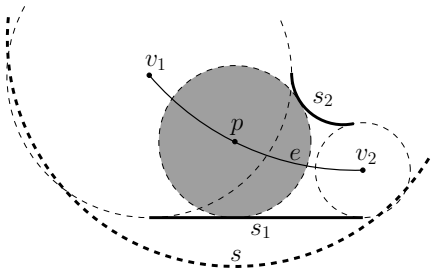


Figure 5: Both nodes v_1, v_2 of the edge e are marked even though some portion of e has to be preserved.

Fortunately, the same strategy that we used in Sec. 2.2.1 to break up a cycle is applicable once more: we consider the normal projection of the center of s onto s_1 and intersect the resulting projection line with e in order to obtain a split point p. Summarizing, we get a uniform strategy for breaking up cycles.

References

[1] H. Alt and O. Schwarzkopf. The Voronoi Diagram of Curved Objects. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 89–97, Vancouver, BC, Canada, 1995.

[2] S. Fortune. A Sweepline Algorithm for Voronoi Diagrams. *Algorithmica*, 2(2):153–174, 1987.

[3] M. Held. VRONI: An Engineering Approach to the Reliable and Efficient Computation of Voronoi Diagrams of Points and Line Segments. *Comput. Geom. Theory and Appl.*, 18(2):95–123, 2001.

[4] K. Sugihara, M. Iri, H. Inagaki, and T. Imai. Topology-Oriented Implementation - An Approach to Robust Geometric Algorithms. *Algorithmica*, 27(1):5–20, 2000.

[5] C. Yap. An $O(n \log n)$ Algorithm for the Voronoi Diagram of a Set of Simple Curve Segments. *Discrete Comput. Geom.*, 2(4):365–393, 1987.

The Entropic Centers of Multivariate Normal Distributions

Frank Nielsen* Richard Nock†

Abstract

In this paper, we seek for a single best representative of a set of statistical multivariate normal distributions. To define the "best" center, we consider either minimizing the average or the maximum relative entropy of the center to the given set of normal distributions. Since the relative entropy is an asymmetric divergence, this yields the notion of left- and right-sided, and symmetrized entropic centroids and circumcenters along with their respective information radii. We show how to instantiate and implement for this special case of multivariate normals very recent work that tackled the broader case of finding centers of point sets with respect to Bregman divergences.

1 Information-theoretic centers

Consider a set of n multivariate normal distributions $\mathcal{D} = \{N(\mu_1, \Sigma_1), ..., N(\mu_n, \Sigma_n)\}$ with $\mu_i \in \mathbb{R}^d$ denoting the mean vector and Σ_i the $d \times d$ symmetric positive semi-definite variance-covariance matrix (i.e., $x^T \Sigma_i x \geq 0$ for all $x \in \mathbb{R}^d$). The probability density function $\Pr(X = x) = p(x; \mu, \Sigma)$ of a normal random variable $X \sim N(\mu, \Sigma)$ is given as:

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{d}{2}} \sqrt{\det \Sigma}} \exp\left(-\frac{(x-\mu)^T \Sigma^{-1} (x-\mu)}{2}\right).$$

A normal distribution $N(\mu, \Sigma)$ can thus be uniquely characterized by a parameter point $\tilde{\Lambda} = (\mu, \Sigma)$ in dimension $D = d + \frac{d(d+1)}{2} = \frac{d(d+3)}{2}$ by stacking the d mean coordinates of μ with the $\frac{d(d+1)}{2}$ matrix coefficients of Σ. We use the tilde notation $\tilde{}$ to emphasize on the *mixed-type* nature vector/matrix of the parameter. For example, bivariate parametric distributions are represented by 5D points lying in the parameter space $\mathcal{X} = \mathbb{R}^2 \times \mathrm{Cone}(\mathbb{R}^{2\times 2})$, where $\mathrm{Cone}(\mathbb{R}^{2\times 2})$ denotes the convex cone of positive semi-definite matrices. Given a set of n d-variate distributions $\mathcal{D}$ handled as D-dimensional point set $\mathcal{S} = \{\tilde{\Lambda}_1, ..., \tilde{\Lambda}_n\}$ with $\tilde{\Lambda}_i = (\mu_i, \Sigma_i)$, we seek to define a proper center. Ignoring for a while the fact that $\mathcal{S}$ is a point set lying in a parameter space $\mathcal{X}$, we may consider the two usual centers in Euclidean geometry $\mathbb{E}^D$: (1) The *centroid* that is commonly called and defined as the center of mass $\bar{\tilde{\Lambda}} = \frac{1}{n} \sum_{i=1}^n \tilde{\Lambda}_i$ of $\mathcal{S}$, and (2) The *circumcenter* $\tilde{\Lambda}^*$ that defines the smallest radius enclosing ball of $\mathcal{S}$.

Both the centroid and the circumcenter are appropriate centers for simplifying the point set down to its best single representative. In other words, these centers solve the 1-clustering task for the following respective minimization criteria:

(1) The *centroid* c^+ is found as the unique minimizer of the minimum average for the *squared* Euclidean distance:

$$c^+ = \arg\min_{x \in \mathbb{R}^D} \sum_{i=1}^n \frac{1}{n} ||x \tilde{\Lambda}_i||^2.$$

(2) The *circumcenter* C^* is defined as the center that minimizes the radius of enclosing balls:

$$C^* = \arg\min_{x \in \mathbb{R}^D} \max_{i=1}^n ||x \tilde{\Lambda}_i||.$$

While these minimization problems look quite similar at first glance, they bear in fact very different mathematical properties. Although it could be tempting to consider "as is" these Euclidean centers for the parameter space Λ, this may not yield meaningful centers properly characterizing well the normal sets. The reason is that the Euclidean distance (or its squared distance) does not make sense[1] for normals. Indeed, consider two univariate normals $X_1 \sim N(\mu, \sigma_1^2)$ and $X_2 \sim N(\mu, \sigma_2^2)$ centered on the *same* mean μ. Applying straightforwardly the Euclidean distance on the parameter points $\lambda_1 = (\mu, \sigma_1^2)$ and $\lambda_2 = (\mu, \sigma_2^2)$, we get a large distance $\sqrt{(\sigma_2^2 - \sigma_1^2)^2}$ for σ_1 deviating much from σ_2, a clearly wrong notion of statistical distance since the bell shape distributions get closer to each other in that case. An appropriate distance between statistical distributions is the *Kullback-Leibler divergence* (KL) better known as *relative entropy*. The KL divergence is an *asymmetric* measure of dissimilarity of probability distributions defined as $\mathrm{KL}(p||q) = \int_x p(x) \log \frac{p(x)}{q(x)} \mathrm{d}x$. The KL divergence relates the Shannon entropy $H(p) = -\int p(x) \log p(x) \mathrm{d}x$ with the cross-entropy $H(p; q) = \int p(x) \log \frac{1}{q(x)} \mathrm{d}x$ as follows: $\mathrm{KL}(p||q) = -H(p) - \int p \log q \mathrm{d}x = H(p; q) - H(p)$. For multivariate normal distributions, the closed-form formula for the entropy and relative entropy are obtained after carrying out fastidious integral computations as $H(p(x; \mu, \Sigma)) = \frac{d}{2} + \frac{1}{2} \log(2\pi)^d \det\Sigma$, (independent of μ) and $\mathrm{KL}(p(x; \mu_i, \Sigma_i)||p(x; \mu_j, \Sigma_j)) = \frac{1}{2}(\mu_i - \mu_j)^T \Sigma_j^{-1} (\mu_i - \mu_j) + \frac{1}{2} \log\det(\Sigma_i^{-1} \Sigma_j) + \frac{1}{2} \mathrm{tr}\left(\Sigma_j^{-1} \Sigma_i\right) - \frac{d}{2}$ (*), where $\mathrm{tr}(\Sigma) = \sum_{i=1}^d \Sigma_{i,i}$ denotes the matrix trace

*Sony Computer Science Laboratories Inc. (FRL) and École Polytechnique (LIX), Frank.Nielsen@acm.org. Web: http://www.sonycsl.co.jp/person/nielsen/

†CEREGMIA, University of Antilles-Guyane, Richard.Nock@martinique.univ-ag.fr

[1] Except iff. all covariance matrices are the same.

(i.e. the sum of the diagonal elements $\Sigma_{i,i}$). Observe that the relative entropy of normals with identity covariance matrices collapses to the *squared* Euclidean distance.

2 Relative entropy of exponential families

It turns out that the normal distributions belong to the (full regular) exponential families [2, 5] in statistics. Besides normal distributions, exponential families include many familiar discrete or continuous distributions such as Poisson, Bernoulli, Beta, Gamma but do not fully cover the spectrum of usual distributions either (e.g., uniform, Lévy SαS or Cauchy distributions). Let λ denote the usual parameters of parametric distributions. Exponential families admit the following canonical decomposition of their probability measures: $p(x;\lambda) = p(x;\theta) = \exp\{< \theta, t(x) > -F(\theta) + C(x)\}$, where $\theta \in \mathbb{R}^D$ are the *natural parameters* associated with the *sufficient statistics* $t(x)$ ($t : \mathbb{R}^d \mapsto \mathbb{R}^D$). The real-valued *log normalizer* function $F(\theta)$ is a strictly convex and differentiable function that specifies uniquely the exponential family, and the function $C(x)$ is the base *counting* or *Lebesgue* measure. Once this canonical decomposition is figured out, we can apply the key *equivalence theorem* [2, 5] Kullback-Leibler of distributions of the *same exponential* family$\Longleftrightarrow$Bregman divergence for the log normalizer F: $\mathrm{KL}(p(x;\mu_i,\Sigma_i)||p(x;\mu_j,\Sigma_j)) = D_F(\theta_j||\theta_i)$, to get without integral computations the closed-form formula (notice that parameter order swaps). The Bregman divergence [2, 5] D_F is defined as the tail of a Taylor expansion for a strictly convex and differentiable function F as $D_F(\theta_j||\theta_i) = F(\theta_j) - F(\theta_i) - < \theta_j - \theta_i, \nabla F(\theta_i) >$, where $< \cdot, \cdot >$ denotes the vector inner product ($< p, q >= p^T q = \sum_{i=1}^d p_i q_i$) and ∇F is the gradient operator. For multivariate normals, we get the *mixed-type* natural parameters $\tilde{\Theta} = (\theta, \Theta) = (\Sigma^{-1}\mu, \frac{1}{2}\Sigma^{-1})$, $F(\tilde{\Theta}) = \frac{1}{4}\mathrm{tr}(\Theta^{-1}\theta\theta^T) - \frac{1}{2}\log\det\Theta + \frac{d}{2}\log 2\pi$ and the one-to-one mapping from the source $\tilde{\Lambda} = (\mu, \Sigma)$ to natural parameters $\tilde{\Theta}$:

$$\tilde{\Lambda} = \begin{pmatrix} \lambda = \mu \\ \Lambda = \Sigma \end{pmatrix} \Longleftrightarrow \tilde{\Theta} = \begin{pmatrix} \theta = \Sigma^{-1}\mu \\ \Theta = \frac{1}{2}\Sigma^{-1} \end{pmatrix}.$$

The inner product $< \tilde{\Theta}_p, \tilde{\Theta}_q >$ in the corresponding Bregman divergence D_F is a *composite* inner product obtained as the sum of two inner products for vectors and matrices: $< \tilde{\Theta}_p, \tilde{\Theta}_q >=< \Theta_p, \Theta_q > + < \theta_p, \theta_q >$. For matrices, the inner product $< \Theta_p, \Theta_q >$ is defined by the trace of the matrix product $\Theta_p\Theta_q^T$: $< \Theta_p, \Theta_q >= \mathrm{Tr}(\Theta_p\Theta_q^T)$. One can check by hand that $\mathrm{KL}(p(x;\mu_i,\Sigma_i)||p(x;\mu_j,\Sigma_j)) = D_F(\tilde{\Theta}_j||\tilde{\Theta}_i)$ yields formula $(*)$ by elementary calculus, bypassing complex integral computations.

3 Legendre transformation and duality

We refer to [2, 5] for detailed explanations that we quickly summarize here: Any Bregman generator function F admits a *dual* Bregman generator function $G = F^*$ via the Legendre transformation $G(y) = \sup_{x\in\mathcal{X}}\{< y, x > -F(x)\}$. The supremum is reached at the *unique* point where the gradient of $G(x) =< y, x > -F(x)$ vanishes, that is when $y = \nabla F(x)$. Writing $\mathcal{X}'_F$ for the *gradient space* $\{x' = \nabla F(x) | x \in \mathcal{X}\}$, the convex conjugate $G = F^*$ of F is the function defined by $F^*(x') =< x, x' > -F(x)$. It follows from Legendre transformation that *any* Bregman divergence D_F admits a *dual* Bregman divergence D_{F^*} related to D_F as follows: $D_F(p||q) = F(p) + F^*(\nabla F(q)) - < p, \nabla F(q) >= F(p) + F^*(q') - < p, q' >= D_{F^*}(q'||p')$. Yoshizawa and Tanabe [10] carried out that non-trivial Legendre transformation for multivariate normals. The strictly convex and differentiable dual Bregman generator function F^* (ie., potential function in information geometry) is $F^*(\tilde{H}) = -\frac{1}{2}\log(1 + \eta^T H^{-1}\eta) - \frac{1}{2}\log\det(-H) - \frac{d}{2}\log(2\pi e)$. The $\tilde{H} \Leftrightarrow \tilde{\Theta}$ coordinate transformations obtained from the Legendre transformation are given by $\tilde{H} = \nabla_{\tilde{\Theta}} F(\tilde{\Theta}) = \begin{pmatrix} \frac{1}{2}\Theta^{-1}\theta \\ -\frac{1}{2}\Theta^{-1} - \frac{1}{4}(\Theta^{-1}\theta)(\Theta^{-1}\theta)^T \end{pmatrix}$, and $\tilde{\Theta} = \nabla_{\tilde{H}} F^*(\tilde{H}) = \begin{pmatrix} -(H + \eta\eta^T)^{-1}\eta \\ -\frac{1}{2}(H + \eta\eta^T)^{-1} \end{pmatrix}$. This yields the dual *expectation* coordinate systems arising from the canonical decomposition:

$$\tilde{H} = \begin{pmatrix} \eta = \mu \\ H = -(\Sigma + \mu\mu^T) \end{pmatrix} \Longleftrightarrow \tilde{\Lambda} = \begin{pmatrix} \lambda = \mu \\ \Lambda = \Sigma \end{pmatrix}.$$

These formulas simplify when we restrict ourselves to diagonal-only covariance matrices Σ_i, spherical Gaussians $\Sigma_i = \sigma_i I$, or univariate normals $\mathcal{N}(\mu_i, \sigma_i^2)$. The expectation parameter $\tilde{H}$ plays an important role for infering the source parameters $\tilde{\Lambda}$ from a sequence of identically and independently distributed observations $x_1, ..., x_v$. Indeed, the maximum likelihood estimator (MLE) of exponential families is $\hat{\tilde{H}} = \frac{1}{v}\sum_{i=1}^v t(x_i)$, where $t(x_i)$ is the sufficient statistics evaluated at x_i. This yields a simple procedure to infer from raw data $x_1, ..., x_v \in \mathbb{R}^d$ the multivariate normal parameters $\hat{\Lambda} \Leftrightarrow \hat{\tilde{H}} \in \mathbb{R}^D$ by taking the centroid on the sufficient statistics for $t(x) = \tilde{x} = (x, -\frac{1}{2}xx^T)$. (This MLE is biased.) Gaussian distribution modeling abound in practice as explicited for three applications in [3].

4 Entropic centroids of multivariate normals

The *entropic centroids* e^+ of normals are defined similarly to the Euclidean geometry case by considering minimizing the *average distance*: the information radius r^+. Because the relative entropy is asymmetric,

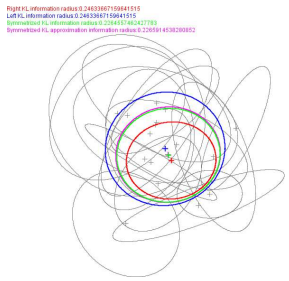


Figure 1: Right-sided (red), left-sided (blue) and symmetrized centroids (green) for 2D normals.

we consider three entropic centroids defined as the following unique minimizers:

$$e_L^+ = \arg\min_{\tilde{\Lambda}} \sum_{i=1}^n \frac{1}{n} \mathrm{KL}(p(x;\tilde{\Lambda})||p(x;\tilde{\Lambda}_i)),$$

$$e_R^+ = \arg\min_{\tilde{\Lambda}} \sum_{i=1}^n \frac{1}{n} \mathrm{KL}(p(x;\tilde{\Lambda}_i||p(x;\tilde{\Lambda})),$$

$$e^* = \arg\min_{\tilde{\Lambda}} \sum_{i=1}^n \frac{1}{2n} \mathrm{KL}(p(x;\tilde{\Lambda}_i||p(x;\tilde{\Lambda})), \\ +\mathrm{KL}(p(x;\tilde{\Lambda})||p(x;\tilde{\Lambda}_i).$$

The latter symmetrical KL divergence is also called J-divergence and plays an important role in signal processing [4]. Using the equivalence theorem $\mathrm{KL} \leftrightarrow D_F$, it follows that the minimizers match up to source↔natural parameter conversions the following *Bregman centroids* c^+ for the log normalizer F by the swapping argument order:

$$e_L^+ \leftrightarrow c_R^+ = \arg\min_{\tilde{\Theta}} \sum_{i=1}^n \frac{1}{n} D_F(\tilde{\Theta}_i||\tilde{\Theta}),$$

$$e_R^+ \leftrightarrow c_L^+ = \arg\min_{\tilde{\Theta}} \sum_{i=1}^n \frac{1}{n} D_F(\tilde{\Theta}||\tilde{\Theta}_i),$$

$$e^+ \leftrightarrow c^+ = \arg\min_{\tilde{\Theta}} \sum_{i=1}^n \frac{1}{n} \frac{D_F(\tilde{\Theta}_i||\tilde{\Theta}) + D_F(\tilde{\Theta}||\tilde{\Theta}_i)}{2}.$$

It has been shown in [7] that the sided Bregman centroids admit *closed-form* formulas that are generalized means[2]: c_R^+ is simply the center of mass $c_R^+ = \bar{\tilde{\Theta}}$ (independent of F, a mean for the identity function) and $c_L^+ = \nabla F^{-1}(\sum_{i=1}^n \nabla F(\tilde{\Theta}_i))$, a ∇F-mean. The information radius [7] r^+ coincides for the sided centroid, and is expressed as a Burbea-Rao divergence (i.e., a generalized F-Jensen reminder): $r^+(\mathcal{S}) = \frac{1}{n}\sum_{i=1}^n F(\tilde{\Theta}_i) - F(\bar{\tilde{\Theta}}) \geq 0$. These results extend to barycenters as well, and allow one to perform interpolation and model merging on statistical normal manifolds [9]. (Note that centroids are robust to outliers.) Further, the *symmetrized entropic centroid* e^+ does not admit closed-form solution but

[2] A f-mean is defined as $f^{-1}(\frac{1}{n}\sum_{i=1}^n f(x_i))$.

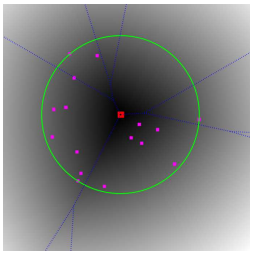


Figure 2: The circumcenter of the smallest enclosing disk is lying on the furthest Voronoi diagram.

is characterized geometrically exactly as the intersection of the geodesic linking the left-sided and right-sided Bregman centroids (say, c_L and c_R respectively) with the mixed-type bisector: $M_F(c_R^F, c_L^F) = \{x \in \mathcal{X} \mid D_F(c_R^F||x) = D_F(x||c_L^F)\}$. This yields an efficient approximation algorithm by walking dichotomically on the geodesic (wrt. the relative entropy) linking the two sided Bregman (c_L^+ and c_R^+) or equivalently entropic centroids (e_L^+ and e_R^+). Figure 1 depicts the sided and symmetrized KL entropic centroids derived from Bregman centroids. The geodesic walk algorithm [7] simplifies and generalizes a former complex and time consuming *ad-hoc method* [4], and allows one to extend the k-means algorithm [2] to hard sided and symmetrized entropic clustering of normals [3]. The Bregman loss function of sided k-means monotonously decreases. (k-means is a Bregman k-means [2] in disguise for the generator $F(x) = x^2$.)

5 Entropic circumcenters of normals

The MinMax optimization problem differs from the MinAvg optimization in the sense that it further optimizes the KL radius r^+ to its smallest possible value r^* but becomes sensitive to outliers. The MinMax optimization problem is *not* differentiable on the furthest Voronoi diagram [5] (see Figure 2). We similarly define the sided and symmetrized entropic circumcenters: $E_L^* \leftrightarrow C_R^* = \arg\min_{\tilde{\Theta}} \max_{i=1}^n D_F(\tilde{\Theta}_i||\tilde{\Theta})$, $E_R^* \leftrightarrow C_L^* = \arg\min_{\tilde{\Theta}} \max_{i=1}^n D_F\tilde{\Theta}||(\tilde{\Theta}_i)$, and $E^* \leftrightarrow C^* = \arg\min_{\tilde{\Theta}} \max_{i=1}^n \frac{D_F(\tilde{\Theta}_i||\tilde{\Theta}) + D_F(\tilde{\Theta}||\tilde{\Theta}_i)}{2}$. We showed that Welzl's MiniBall algorithm extends to arbitrary Bregman divergences [8] allowing us to compute exactly the sided entropic circumcenters E_R^* and E_L^* on the plane (see Figure 3). The basis computations relies on using the fact that the *right-sided Bregman Voronoi bisector* [5] is a straight line. (In small dimensions, computing these bases proceed by dimension reduction by constructing the Bregman generator restricted to affine subspaces.) Note that the circumcenter of the entropic ball passing through exactly three points may not exist as this circumcenter obtained as the common intersection point of three linear bisectors may potentially fall out of domain $\mathcal{X}$.

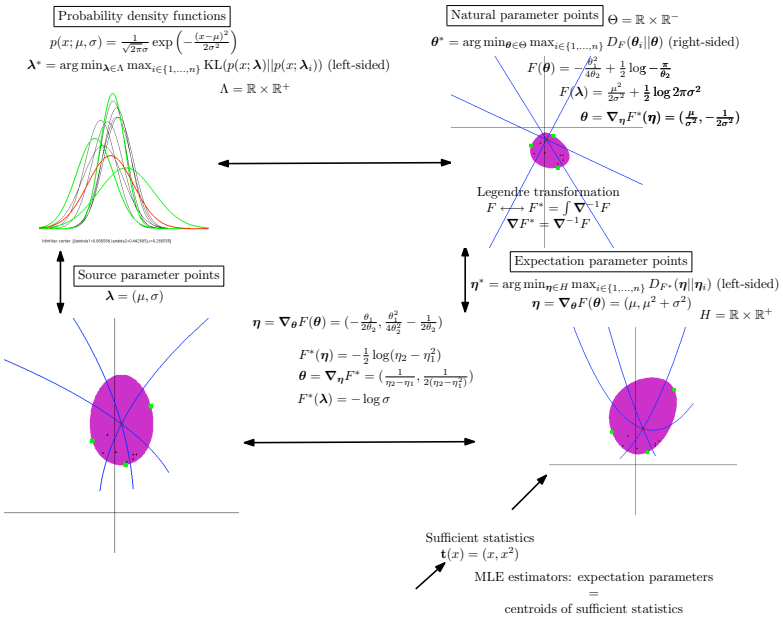


Figure 3: The left KL circumcenter of a set of 1D normals.

However, this never happens for the recursive generalization of Welzl's algorithm [8]. As dimension increases, it is not possible to compute in practice the exact circumcenter as Welzl's algorithm exhibits the *curse of dimensionality*: an exponential time dependence with the dimension. We considered in [6] a generalization of the approximation of the smallest enclosing ball based on the notion of *core-sets* working in very large dimensions ($d \sim 10000$). As mentioned above, the computation of the smallest enclosing entropic balls rely on the property that right-type Bregman bisector are hyperplanes [5], and therefore the right-type Bregman Voronoi is an affine diagram that can be computed equivalently using a power diagram [5]. This allows us to define *entropic Voronoi diagrams* for multivariate normals with corresponding dual regular/geodesic entropic Delaunay triangulations.

6 Concluding remarks

We have concisely presented in view of our results on information-theoretic Bregman centers [7, 8, 6] the entropic centers of statistical multivariate normal distributions, i.e. the Kullback-Leibler entropic centroids and circumcenters. We have described the Legendre transformation for the mixed-type vector/matrix log normalizer of that exponential family and reported on our implementation. We can reinterpret these entropic centers under the auspices of information geometry [1] by considering the dually flat Riemannian manifolds where Bregman divergences arise naturally as the canonical divergences [10].

References

[1] S.-I. Amari and H. Nagaoka. *Methods of Information Geometry*. Oxford University Press, 2000. ISBN-10:0821805312.

[2] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh. Clustering with Bregman divergences. *Journal of Machine Learning Research*, 6:1705–1749, 2005.

[3] J. V. Davis and I. S. Dhillon. In *Neural Information Processing Systems*, pages 337–344, 2006.

[4] T. A. Myrvoll and F. K. Soong. On divergence-based clustering of normal distributions and its application to HMM adaptation. In *EuroSpeech*, volume 2, pages 1517–1520, 2003.

[5] F. Nielsen, J.-D. Boissonnat, and R. Nock. Bregman Voronoi diagrams: Properties, algorithms and applications, 2007. arXiv.org:0709.2196 (Extend SODA'07).

[6] F. Nielsen and R. Nock. On approximating the smallest enclosing Bregman balls. In *Proc. 22nd Symposium on Computational Geometry*, pages 485–486, 2006.

[7] F. Nielsen and R. Nock. On the centroids of symmetrized Bregman divergences. 2007. arXiv.org:0711.3242.

[8] F. Nielsen and R. Nock. On the smallest enclosing information disk. *Information Processing Letters*, 105(3):93–97, 2008.

[9] B. Pelletier. Informative barycentres in statistics. *Annals of the Institute of Statistical Mathematics*, 57(4):767–780, 2005.

[10] S. Yoshizawa and K. Tanabe. Dual differential geometry associated with Kullback-Leibler information on the Gaussian distributions and its 2-parameter deformations. *SUT Journal of Mathematics*, 35(1):113–137, 1999.

Quantum Voronoi Diagrams

Frank Nielsen* Richard Nock†

Abstract

We introduce the smooth parametric family of Bregman-Csiszár quantum entropies including the usual von Neumann and Burg quantum entropies. We then describe the dualistic nature of Voronoi diagrams for 1-qubit quantum states inside the 3D Bloch ball representation. We show that these diagrams can be computed as Bregman Voronoi diagrams for the corresponding entropic Bregman generator acting on Hermitian density matrices. This implies that these dual diagrams can be derived from equivalent power diagrams of balls in the Laguerre geometry, and allows one to prove by extension that the von Neumann quantum Voronoi diagram on the degenerated Bloch sphere of pure quantum states coincides with the ordinary Euclidean Voronoi diagram, bypassing the fact that the quantum divergence is not properly defined there.

1 Introduction and preliminaries

The 21st century attests the accelerated rise of the deployment of quantum mechanics into various industrial prototypes like the prominent quantum cryptographic systems. Recent breakthroughs in experimental physics bridged the gap between mathematical theory and practice, and the analysis of quantum channel characteristics such as its capacity become a fundamental problem associated with related open problems.[1] In quantum information theory [1], particle state distributions are analyzed probabilistically by means of *density* matrices $\mathbf{X}$. A d-level system is characterized by a $d \times d$ matrix $\mathbf{X} \in \mathbb{C}_{d\times d}$ with complex coefficients that satisfies the following three properties:

1. $\mathbf{X}$ is Hermitian. That is, $\mathbf{X}$ is equal to its *conjugate transpose*: $\mathbf{X} = \mathbf{X}^{*T}$,
2. $\mathbf{X}$ has *unit trace*. That is, the sum of diagonal elements sums up to one and has no imaginary part: $\mathrm{Tr}(\mathbf{X}) = \sum_{i=1}^{d} X_{i,i} = 1$,
3. $\mathbf{X}$ is *semi-positive definite*. That is, $\mathbf{X}$ belongs to the positive cone: $\mathbf{x}^T\mathbf{X}\mathbf{x} \geq 0\ \forall \mathbf{x} \neq \mathbf{0}$. This condition is equivalent to non-negative determinant: $\det \mathbf{X} \geq 0$.

Let $\mathcal{S}(\mathbb{C}^d)$ denote the space of such semi-positive definite density matrices of size $d \times d$. One qubit (quantum bit) systems[2] are the simplest fundamental case obtained for $d = 2$. The above three conditions imply the following family of 2×2 complex matrices:

$$\mathbf{X} = \left\{ \frac{1}{2} \begin{bmatrix} 1+z & x-iy \\ x+iy & 1-z \end{bmatrix} \mid x^2+y^2+z^2 \leq 1 \right\},$$

where i denotes the imaginary number $i^2 = -1$. The condition $x^2 + y^2 + z^2 \leq 1$ is derived from the semi-positive definiteness assumption ($\det \mathbf{X} \geq 0$). Thus 1-qubit states $\mathbf{X}$ can be represented equivalently by a triple $\mathbf{x} = (x, y, z)$ of reals, a 3D point $\mathbf{x} \in \mathbb{R}^3$, and the set $\mathcal{S}(\mathbb{C}^2)$ of 1-qubits is referred to as the *Bloch ball*[3]. We distinguish between *pure* states which have degenerated density matrices of rank 1 (noninvertible matrix), and *mixed* states of full rank 2. The pure state condition is geometrically visualized by density matrices lying on the boundary of the Bloch ball: The *Bloch sphere*. The state $\mathbf{X}$ of a 1-qubit is expressed using three reals $\mathbf{x} = (x, y, z)$ that can be reinterpreted using spherical coordinates as $\mathbf{x} = (r, \theta, \phi)$ where r denotes the *radius* of the state to the origin, and θ and ϕ encode the *latitude* and *longitude* rotation angles:

$$(r, \theta, \phi) \leftrightarrow \begin{bmatrix} x = r\sin\theta\cos\phi \\ y = r\sin\phi \\ z = r\cos\theta\cos\phi \end{bmatrix}.$$

In order to define the *quantum divergence* that is the generalization of the Kullback-Leibler divergence (better known as relative entropy [1]) to density matrices, we first define the *logarithm* of a density matrix using its *spectral* decomposition. Consider the singular value decomposition (SVD) of a Hermitian matrix $\mathbf{X}$: $\mathbf{X} = \mathbf{V}\ \mathrm{Diag}(\boldsymbol{\lambda})\ \mathbf{V}^*$, with both $\mathbf{V}$ and $\mathbf{V}^*$ unitary orthonormal matrices, and all eigenvalues $\lambda_i \geq 0$ real and positive. The diagonal matrix represents the *eigenspectra* and the complex orthonormal rotation matrix $\mathbf{V}$ the associated *eigenspace*. Using the spectrum decomposition

*Sony Computer Science Laboratories Inc. (FRL) and École Polytechnique (LIX), Frank.Nielsen@acm.org. Web: http://www.sonycsl.co.jp/person/nielsen/

†CEREGMIA, University of Antilles-Guyane, Richard.Nock@martinique.univ-ag.fr

[1] See the quantum information problems list at http://www.imaph.tu-bs.de/qi/problems/

[2] In general, n-qubit systems require dimension $d = 2^n$.

[3] Named after physicist Felix Bloch, first director of the CERN institute.

of a matrix $\mathbf{X}$, we define the logarithm of a density matrix as: $\log \mathbf{X} = \mathbf{V}\ \mathrm{Diag}(\log \lambda_1, ..., \log \lambda_d)\ \mathbf{V}^*$. The *quantum von Neuman entropy* $H(\mathbf{X})$ (matrix entropy) is a generalization of the classical *Shannon entropy*[4] to density matrices: $H(\mathbf{X}) = -\mathrm{Tr}(\mathbf{X}\log\mathbf{X})$. It can be shown that the quantum entropy is equal to the Shannon entropy for the eigenvalue distribution: $H(\mathbf{X}) = H(\boldsymbol{\lambda}) = -\sum_{i=1}^{d}\lambda_i \log\lambda_i$. The quantum information divergence I (matrix relative entropy) generalizes the Kullback-Leibler divergence (KL) by considering the following distortion measure: $I(\mathbf{P}||\mathbf{Q}) = \mathrm{Tr}\left(\mathbf{P}(\log\mathbf{P} - \log\mathbf{Q})\right) \geq 0$. This divergence is *not* symmetric nor does it satisfy the triangle inequality. It is therefore not a metric. Note that the quantum divergence is defined for $\mathbf{P} \to \mathbf{0}$ by taking the limit: $\lim_{\mathbf{X}\to\mathbf{0}} \mathrm{Tr}(\mathbf{X}\log\mathbf{X}) = 0$ (since $\lim_{x\to 0} x\log x = 0$). However, the divergence is *not* properly defined when $\mathbf{Q}$ is *not* full rank (i.e., $\mathbf{Q}$ encodes a pure state) because of the undefined logarithm. The quantum information divergence is *reflexive*: $I(\mathbf{P}||\mathbf{Q}) = 0 \Leftrightarrow \mathbf{P} = \mathbf{Q}$. We further have the following quantum/classical information inequality: $I(\mathbf{P}||\mathbf{Q}) \geq \mathrm{KL}(\boldsymbol{\lambda}_\mathbf{P}||\boldsymbol{\lambda}_\mathbf{Q}) \geq 0$, where $\boldsymbol{\lambda}_\mathbf{P}$ and $\boldsymbol{\lambda}_\mathbf{Q}$ are the eigenvalue distributions of the spectral decomposition of the density matrices $\mathbf{P}$ and $\mathbf{Q}$, respectively. The inequality is *strict* if and only if the eigenspaces of $\mathbf{P}$ and $\mathbf{Q}$ differ. Interestingly, this von Neumann quantum information divergence belongs to a broader class of parametric divergences called Bregman divergences [2]. Bregman divergences are parameterized families of distortion measures induced by a strictly convex and differentiable convex function $F : \mathcal{S}(\mathbb{C}^d) \to \mathbb{R}$ such that:

$$D_F(\mathbf{P}||\mathbf{Q}) = F(\mathbf{P}) - F(\mathbf{Q}) - \langle \mathbf{P}-\mathbf{Q}, \boldsymbol{\nabla} F(\mathbf{Q})\rangle \quad (1)$$

where the inner product is defined as: $\langle \mathbf{P}, \mathbf{Q}\rangle = \mathrm{Tr}(\mathbf{P}\mathbf{Q}^*)$, and $\boldsymbol{\nabla} F(\cdot)$ is the Gâteaux derivative: The gradient. We have $D_F(\mathbf{P}||\mathbf{Q}) = 0$ if and only if $\mathbf{P} = \mathbf{Q}$ (positive-definiteness generalizing Gibb's inequality). Bregman divergences can also be interpreted *locally* as quadratic distance measures by considering the Taylor expansion of F with an *exact remainder* term: $D_F(\mathbf{P}||\mathbf{Q}) = (\mathbf{P}-\mathbf{Q})^* \frac{\boldsymbol{\nabla} F^2(\boldsymbol{\varepsilon})}{2}(\mathbf{P}-\mathbf{Q})$, where $\boldsymbol{\varepsilon}$ depends on both $\mathbf{P}$ and $\mathbf{Q}$. The quantum information divergence is a Bregman divergence obtained for the Bregman generator $F(\mathbf{X}) = \mathrm{Tr}(\mathbf{X}\log\mathbf{X})$. Dhillon and Tropp [3] thoroughly investigated Bregman matrix distortion measures for "matrix nearness" decompositions with a special care given to the squared Fröbenius, von Neumann information and the log det divergences obtained respectively for the generators $F(\mathbf{X}) = \frac{1}{2}||\mathbf{X}||^2$, $F(\mathbf{X}) = \mathrm{Tr}(\mathbf{X}\log\mathbf{X})$ and $F(\mathbf{X}) = -\log\det\mathbf{X}$. It follows that the quantum relative entropy $I = D_F$ (von Neumann Bregman divergence) has thus a neat axiomatic characterization [4], and can further be *extended* following Csiszár least square projection characterization [4], by using for the Bregman generator the *extended negative entropy* $F(\mathbf{X}) = \mathrm{Tr}(\mathbf{X}\log\mathbf{X} - \mathbf{X})$. The gradient of $F(\mathbf{X})$ is $\boldsymbol{\nabla} F(\mathbf{X}) = \log\mathbf{X}$, the quantum Burg entropy. In summary, the extended von Neumann quantum divergence is a Bregman divergence in disguise for generator $F(\mathbf{X}) = \mathrm{Tr}(\mathbf{X}\log\mathbf{X} - \mathbf{X})$: $I(\mathbf{P}||\mathbf{Q}) = D_F(\mathbf{P}||\mathbf{Q}) = \mathrm{Tr}\left(\mathbf{P}(\log\mathbf{P} - \log\mathbf{Q}) - \mathbf{P} + \mathbf{Q}\right)$. Bregman divergences are invariant by affine terms and enjoy a remarkable bijection with probability distributions of the statistical exponential families [5]. Carrying out the calculations for the 3D Bloch ball representation of 1-qubits, we obtain [7]:

$$\begin{aligned} I(\mathbf{P}||\mathbf{Q}) &= \frac{1+r_P}{2}\log\frac{1+r_P}{2} + \frac{1-r_P}{2}\log\frac{1-r_P}{2} \\ &\quad -\frac{1}{2}\log\frac{1-r_Q^2}{4} - \frac{<\mathbf{p},\mathbf{q}>}{2r_Q}\log\frac{1+r_Q}{1-r_Q} \quad (2) \\ &\stackrel{\mathrm{def}}{=} a(r_P) + b(r_Q) - c(r_Q) <\mathbf{p},\mathbf{q}>. \quad (3) \end{aligned}$$

2 Quantum Bregman-Csiszár divergences

The choice of the proper quantum divergence may depend upon the situation implied by the underlying study or application needs [1]. It is therefore interesting to design a *flexible generic* quantum divergence by generalizing *parametric* divergences proposed in classical information theory. We propose to extend the von Neumann quantum/Log det divergences [3] to the class of *quantum Bregman-Csiszár divergences* for density matrices. First, define the smooth family of strictly convex and differentiable Bregman generators F_α on density matrices for a single parameter $\alpha \in [0,1]$ as: $F_\alpha(\mathbf{X}) = \frac{1}{\alpha(1-\alpha)}\mathrm{Tr}\left(-\mathbf{X}^\alpha + \alpha\mathbf{X} - \alpha\mathbf{I} + \mathbf{I}\right)$, where $\mathbf{I}$ denotes the identity matrix, and $\mathbf{X}^\alpha$ is defined from the spectral decomposition of $\mathbf{X} = \mathbf{V}\ \mathrm{Diag}(\boldsymbol{\lambda})\ \mathbf{V}^*$ as $\mathbf{X}^\alpha = \mathbf{V}\ \mathrm{Diag}(\lambda_1^\alpha, ..., \lambda_d^\alpha)\ \mathbf{V}^*$. It follows from the Bregman divergence of Eq. 1 the α-quantum Bregman divergence: $D_\alpha(\mathbf{P}||\mathbf{Q}) = \frac{1}{\alpha(1-\alpha)}\mathrm{Tr}\left(\mathbf{Q}^\alpha - \mathbf{P}^\alpha + \alpha\mathbf{Q}^{\alpha-1}(\mathbf{P}-\mathbf{Q})\right)$. Note that since Bregman generators are equivalent up to affine terms [8], we find in the limit case that F_0 is the quantum Burg entropy and F_1 is the usual von Neumann entropy. $F_\alpha(\mathbf{X})$ is defined on both mixed and pure states for $\alpha \in (0,1)$. Further, notice that we have in the limit case $\lim_{\alpha\to 1} F_\alpha(\mathbf{X}) = F(\mathbf{X}) = \mathrm{Tr}(\mathbf{X}\log\mathbf{X} - \mathbf{X})$ and $\lim_{\alpha\to 1}\boldsymbol{\nabla} F_\alpha(\mathbf{X}) = \lim_{\alpha\to 1}\frac{1}{1-\alpha}(\mathbf{I} - \mathbf{X}^{\alpha-1}) = \log\mathbf{X}$.

3 Quantum Voronoi diagrams

Kato et al. [6, 7] studied the Voronoi diagram for 1-qubit systems with respect to the Fubini-Study D_{FS} and Bures D_{B} metric distances for pure states. Let $\mathbf{P}$

[4] The Shannon entropy of a discrete d-dimensional distribution $\mathbf{p}$ is defined as $H(\mathbf{p}) = -\sum_{i=1}^{d} p_i \log p_i$.

Figure 1: Quantum Voronoi diagram of 1-qubit pure states: Voronoi cells annotated with density matrices on the latitude-longitude map (left) and Voronoi cells visualized on the 3D Bloch sphere (right).

and $\mathbf{Q}$ denote two pure state density matrices represented on the Bloch sphere by 3D points $\mathbf{p}$ and $\mathbf{q}$, respectively. We have $D_{\mathrm{FS}}(\mathbf{P},\mathbf{Q}) = \arccos\sqrt{\mathrm{Tr}(\mathbf{PQ})} = \arccos\sqrt{\frac{1+<\mathbf{p},\mathbf{q}>}{2}}$ and $D_{\mathrm{B}}(\mathbf{P},\mathbf{Q}) = \sqrt{1-\mathrm{Tr}(\mathbf{PQ})} = \frac{1}{\sqrt{2}}||\mathbf{p}-\mathbf{q}||$. Observe that we just need the diagonal elements[5] selected by the trace operator for computing these quantum distances. Kato et al. [6, 7] thus show that for the case of pure state 1-qubits, these Voronoi diagrams are *equivalent* to the ordinary Voronoi diagram on the sphere. This spherical Voronoi diagram can in turn be simply obtained as the ordinary 3D Euclidean Voronoi diagram restricted to the unit (Bloch) sphere. Figure 1 depicts such a quantum Voronoi diagram under these metrics with density matrix annotations in each Voronoi cell. Moreover, Kato et al. [6, 7] investigated the Voronoi diagram with respect to the quantum information divergence and carried out calculations in the limit case of $\mathbf{Q}$ being a pure state. They deduce that the quantum Voronoi diagram of pure states is identical to the conventional spherical Voronoi diagram although they *differ* for mixed states. We revisit concisely these results under the framework of Bregman Voronoi diagrams [8] and show how to naturally extend these diagrams to pure states from corresponding affine power diagrams *fully* defined over $\mathbb{R}^3$. Since Bregman divergences are usually asymmetric [8], we consider the *left-sided* and *right-sided* Bregman Voronoi diagrams of density matrix set $\mathcal{P} = \{\mathbf{P}_1, ..., \mathbf{P}_n\}$ defined as the cell complex induced by the left- and right-sided *Bregman bisectors*. The Voronoi cells $\mathrm{Vor}_F(\mathbf{P}_i) = \cap_{j\neq i} H_F^+(\mathbf{P},\mathbf{P}_j)$ and $\mathrm{Vor}'_F(\mathbf{P}_i) = \cap_{i\neq j} H_F'^+(\mathbf{P},\mathbf{P}_j)$ are defined respectively as intersections of influence regions bounded using the corresponding sided bisectors H_F and $H_{F'}$ as follows: $H_F(\mathbf{P},\mathbf{Q}) = \{\mathbf{X}|\ D_F(\mathbf{X}||\mathbf{P}) = D_F(\mathbf{X}||\mathbf{Q})\}$, and $H'_F(\mathbf{P},\mathbf{Q}) = \{\mathbf{X}\mid D_F(\mathbf{P}||\mathbf{X}) = D_F(\mathbf{Q}||\mathbf{X})\}$.

These bisectors match only for symmetric Bregman divergences that are generalized quadratic distances [8]. Let $\mathbf{X}' = \boldsymbol{\nabla} F(\mathbf{X})$ denote the gradient of Hermitian matrix $\mathbf{X}$. The left-sided bisector is always a *hyperplane* [8] whatever the considered generator F: $H_F(\mathbf{P},\mathbf{Q}) : \langle \mathbf{X}, \mathbf{P}'-\mathbf{Q}'\rangle + F(\mathbf{P}) - \langle \mathbf{P},\mathbf{P}'\rangle - F(\mathbf{Q}) + \langle \mathbf{Q},\mathbf{Q}'\rangle = 0$. This equation becomes for the case of the *extended* negative von Neuman entropy (with $\mathbf{P}' = \log\mathbf{P}$), the following hyperplane equation in dimension d: $H_F(\mathbf{P},\mathbf{Q}) : \{\mathbf{X} \mid \mathrm{Tr}\left(\mathbf{X}(\log\mathbf{P} - \log\mathbf{Q}) - \mathbf{P} + \mathbf{Q}\right) = 0\}$. Using the a, b, c notations of the spherical coordinates of Eq. 3, it follows that for 1-qubit states on the 3D Bloch ball we have the bisector *plane* equation: $H_F(\mathbf{p},\mathbf{q}) : \{\mathbf{x}| < \mathbf{x}, c(r_Q)\mathbf{q} - c(r_P)\mathbf{p} > + b(r_P) - b(r_Q) = 0\}$. Next, we show that the right-type bisector is not linear but dually linear in the gradient space $\boldsymbol{\nabla} F(\mathbf{X})$.

3.1 Legendre duality and bisectors

Since F is strictly convex and differentiable, we associate to F a unique dual conjugate function F^* via the Legendre-Fenchel slope transformation such that: $F^*(\mathbf{Y}) = \sup_{\mathbf{X}\in\mathcal{S}(\mathbb{C}^d)}\{\langle \mathbf{Y},\mathbf{X}\rangle - F(\mathbf{X})\}$. The unique supremum is reached at point $\mathbf{Y} = \boldsymbol{\nabla} F(\mathbf{X}) = \mathbf{X}'$. The Legendre transformation defines a *dual* quantum Bregman divergence for the dual generator $F^*(\mathbf{X}) = \mathrm{Tr}(\exp\mathbf{X})$ ($\exp\mathbf{X}$ is again defined using the spectral decomposition $\exp\mathbf{X} = \mathbf{V}\ \mathrm{Diag}(\exp\lambda_1, ..., \exp\lambda_d)\ \mathbf{V}^*$): $D_F(\mathbf{P}||\mathbf{Q}) = F(\mathbf{P}) + F^*(\mathbf{Q}') - \langle \mathbf{P},\mathbf{Q}'\rangle = D_{F^*}(\mathbf{Q}'||\mathbf{P}')$. Thus although the right-sided bisector is not linear, it is dually linear by considering the associated dual Bregman generator: $H'_F(\mathbf{P},\mathbf{Q}) \equiv H_{F*}(\mathbf{Q}',\mathbf{P}')$.

For 1-qubit states represented by a 3D point inside the Bloch ball, we have the Legendre conjugate explicited using 3D coordinates as [6]: $(x^*, y^*, z^*) = \nabla F_B(x,y,z) = \frac{1}{2r}\log\frac{1+r}{1-r}(x,y,z)$, where ∇F_B is the gradient for the 3D Bloch generator function, and r is the radius $r = \sqrt{x^2+y^2+z^2}$. Observe that the dual Legendre function ${F_B}^* = \int \nabla F_B^{-1}$ does not admit any closed-form formula although we can easily tabulate it in practice for fine approximations by using a 1D look-up-table array.

3.2 Affine parametric quantum Voronoi diagrams

Since the left-type Voronoi diagram is affine, we use the handy *universal construction* of affine diagram from power diagram [9] to define quantum Voronoi diagrams as power diagrams of balls in the Laguerre geometry. We associate to density matrix $\mathbf{X}_i$ the ball with Hermitian matrix center:

$$\boldsymbol{\nabla} F_\alpha(\mathbf{X}_i) = \frac{1}{1-\alpha}\left(\mathbf{I} - \mathbf{V}_i \begin{bmatrix} \lambda_{i,1}^{\alpha-1} & 0 \\ 0 & \lambda_{i,2}^{\alpha-1} \end{bmatrix} \mathbf{V}_i^*\right) \quad (4)$$

$$\text{with } \mathbf{V}_i = \frac{1}{\sqrt{2}}\begin{bmatrix} \frac{x_i - iy_i}{\sqrt{x_i^2+y_i^2}}\sqrt{\frac{r_i+z_i}{r_i}} & \frac{x_i-iy_i}{\sqrt{x_i^2+y_i^2}}\sqrt{\frac{r_i-z_i}{r_i}} \\ \sqrt{\frac{r_i-z_i}{r_i}} & -\sqrt{\frac{r_i+z_i}{r_i}} \end{bmatrix}$$

[5] In general, for computing the quantum divergence between any two $d \times d$ states $\mathbf{P}$ and $\mathbf{Q}$, we just need $O(d^2)$ operations for computing the matrix product of density matrices along the diagonal only.

and squared radius [8]

$$r_i^2 = \langle \nabla F_\alpha(\mathbf{X}_i), \nabla F_\alpha(\mathbf{X}_i) \rangle + 2(F_\alpha(\mathbf{X}_i) - \langle \mathbf{X}_i, \nabla F_\alpha(\mathbf{X}_i) \rangle), \quad (5)$$

that is potentially imaginary and infinite in the limit case $\alpha \to 1$ for pure states since one eigenvalue $\frac{1-r}{2}$ is zero [6]. The power bisector of two 3D Euclidean balls $B(\mathbf{p}, r_P)$ and $B(\mathbf{q}, r_Q)$ centered at 3D points $\mathbf{p}$ and $\mathbf{q}$ is the radical hyperplane of equation [8]: $2 < \mathbf{x}, \mathbf{q} - \mathbf{p} > + < \mathbf{p}, \mathbf{p} > - < \mathbf{q}, \mathbf{q} > + r_Q^2 - r_P^2 = 0$. Since pure states have the *same* equivalent ball radius (being infinite in this limit case, see Eq. 5) it follows that the dual quantum Voronoi diagrams on the Bloch sphere has *in the limit case* (proof omitted) matching affine bisectors which coincides exactly with the bisector equation[6] for the 3D ordinary Euclidean Voronoi diagram for 3D points $\mathbf{p}$ and $\mathbf{q}$ on the Bloch sphere.

Theorem 1 *The α-entropic quantum Voronoi diagrams are Bregman Voronoi diagrams that can be computed from equivalent power diagrams. In the limit case $\alpha \to 1$, the von Neumann quantum Voronoi diagrams on the Bloch sphere of pure states coincides with an ordinary Euclidean Voronoi diagram restricted to the sphere.*

This reduction to power diagrams is attractive since power diagrams are defined on the *full Euclidean space* $\mathbb{E}^3$ (i.e., inside mixed states, on pure states and outside the Bloch sphere). Figure 2 depicts an example of equivalence of the quantum Voronoi↔power Voronoi diagrams for quantum states expressed in the *same eigenspace* (i.e., a 2D slice of the 3D quantum Voronoi diagram that is also equivalent to a 2D power diagram). In that case, the sliced quantum Voronoi diagram is a classical extended Kullback-Leibler Voronoi diagram on the eigenvalues. The 1-qubit quantum Voronoi diagram can be computed easily using 3D power diagrams [9] and $\mathbf{X} \leftrightarrow \nabla F_\alpha(\mathbf{X})$ conversions [8]. Although the left-side and right-side quantum Voronoi diagrams on pure states match and coincide with the Euclidean Voronoi diagram for $\alpha = 1$, it is not anymore the case *inside* the Bloch ball of mixed states where they provably differ [8]. As an application, notice that the Holevo capacity of a quantum channel [10] can be computed from the furthest quantum Voronoi diagram [6].

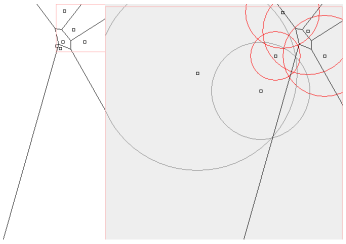


Figure 2: Sliced quantum Voronoi diagram (fixed eigenspace, left) equivalent to a 2D power diagram (right) in Euclidean geometry.

References

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[2] L. M. Bregman, "The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming," *USSR Computational Mathematics and Mathematical Physics*, vol. 7, pp. 200–217, 1967.

[3] I. S. Dhillon and J. A. Tropp, "Matrix nearness problems with Bregman divergences," *SIAM Journal on Matrix Analysis and Applications*, vol. 29, no. 4, pp. 1120–1146, 2007.

[4] I. Csiszar, "Why least squares and maximum entropy? an axiomatic approach to inference for linear inverse problems," *The Annals of Statistics*, vol. 19, no. 4, pp. 2032–2066, 1991.

[5] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh, "Clustering with Bregman divergences," *Journal of Machine Learning Research (JMLR)*, vol. 6, pp. 1705–1749, 2005.

[6] K. Kato, M. Oto, H. Imai, and K. Imai, "Voronoi diagrams for pure 1-qubit quantum states,quant-ph/0604101, 2006.

[7] K. Kato, H. Imai and K. Imai, *Error Analysis of a Numerical Calculation about One-qubit Quantum Channel Capacity*, ISVD, pp. 265-269, 2007.

[8] F. Nielsen, J.-D. Boissonnat, and R. Nock, "Bregman Voronoi diagrams: Properties, algorithms and applications," 2007, arXiv.org:0709.2196 (ACM-SIAM SODA'07).

[9] F. Aurenhammer, "Power diagrams: properties, algorithms and applications," *SIAM J. Comput.*, vol. 16, no. 1, pp. 78–96, 1987.

[10] A. S. Holevo, "The capacity of quantum channel with general signal states," *IEEE Transactions on Information Theory*, vol. 44, quant-ph/9611023, 1998.

[6] Indeed, we have in Euclidean geometry $||\mathbf{xp}|| = ||\mathbf{xq}|| \Leftrightarrow < \mathbf{x} - \mathbf{p}, \mathbf{x} - \mathbf{p} > = < \mathbf{x} - \mathbf{q}, \mathbf{x} - \mathbf{q} >$ (by squaring norms) from which we get the ordinary bisector equation: $2 < \mathbf{x}, \mathbf{q} - \mathbf{p} > + < \mathbf{p}, \mathbf{p} > - < \mathbf{q}, \mathbf{q} > = 0$.

Triangulating the 3D Periodic Space*

Manuel Caroli † Nico Kruithof ‡ Monique Teillaud §

Abstract

In this extended abstract we discuss the computation of Delaunay triangulations of point sets in the 3-dimensional periodic space $\mathbb{T}^3$. We reuse the standard incremental algorithm for computing Delaunay triangulations in $\mathbb{R}^3$ and describe how to adapt it to compute in $\mathbb{T}^3$. We present solutions to several problems we encountered as well as some basic ideas of the implementation and optimizations.

1 Introduction

Computing Delaunay triangulations of point sets is a well-studied subject in Computational Geometry. There are many algorithms available [4, 1] as well as implementations [3, 12, 7, 9]. However, these algorithms are usually restricted to triangulations in $\mathbb{R}^d$. The goal of our work is to extend the CGAL 3D triangulation package [8] to compute triangulations in 3-dimensional spaces other than $\mathbb{R}^3$. We explore here the periodic space $\mathbb{T}^3$. There are several applications of Delaunay triangulations in this space, particularly in simulation [10], but up to now there has not been an implementation available. Usually, heuristic approaches duplicating points close to the boundary and computing $\mathbb{R}^3$ triangulations of these modified point sets have been used. However, this is inefficient and gives only an approximation of the triangulation.

In the following we first review triangulations, then we examine $\mathbb{T}^3$. Next, we present our solutions to several problems we encountered regarding both correctness and efficiency. Finally, we give benchmarks comparing the triangulations of $\mathbb{R}^3$ and $\mathbb{T}^3$ and measuring the impact of some optimizations. More details about implementation and software design can be found in [2].

2 Triangulations

Given a set $\mathcal{S}$ of points in $\mathbb{R}^d$, a triangulation partitions the space into cells (tetrahedra in 3D) whose vertices are the given points. The *Delaunay triangulation* has the property that the circumscribing ball of each cell does not contain any other point of $\mathcal{S}$ [1, 4].

To give a more precise definition of a triangulation we need to recall the notion of a simplicial complex first. More details can be found in [14, 11].

Definition 1

- *A* k-simplex *is the convex hull of a set of* $k+1$ *affinely independent points. A 0-simplex is called a* vertex.
- *If* σ *is a simplex defined by a finite point set* $\mathcal{S}$*, then any simplex* τ *defined by* $\mathcal{T} \subset \mathcal{S}$ *is called a* face *of* σ.
- *A* simplicial complex K *is a finite collection of non-empty simplices such that the following two conditions hold:*
  1. *if* $\sigma \in K$ *and* τ *is a face of* σ*, then* $\tau \in K$,
  2. *if* $\sigma_1, \sigma_2 \in K$*, then their intersection* $\sigma_1 \cap \sigma_2$ *is either empty or a face of both* σ_1 *and* σ_2.

For $0 \leq k \leq 3$ we use the following terms to denote the respective k-simplices: *vertex, edge, facet, cell*. Now we can define a triangulation as follows:

Definition 2 *Let* $\mathcal{S}$ *be a finite point set in some space* $\mathbb{X}$*. A simplicial complex* K *is a* triangulation *of* $\mathcal{S}$*, if*
1. *each point in* $\mathcal{S}$ *is a vertex of* K,
2. $\bigcup_{\sigma \in K} \sigma$ *is homeomorphic to* $\mathbb{X}$.

3 The periodic space $\mathbb{T}^3$

Topologically, $\mathbb{T}^3$ is the hypersurface of a torus in 4D. Let S^1 denote the 1-dimensional sphere, then $\mathbb{T}^3$ is generated by $S^1 \times S^1 \times S^1$. Therefore we can assume a coordinate system in the torus with coordinates in $[0,1)^3$. We denote points in $\mathbb{R}^3$ by $p = (x, y, z)$ and points in $\mathbb{T}^3$ by $q = (u, v, w)$. We map $\mathbb{T}^3$ onto $\mathbb{R}^3$ in the following way:

$$g(q) := \{(u, v, w) + (i, j, k) \mid i, j, k \in \mathbb{Z}\}$$

Intuitively, we repeat $[0,1)^3$ infinitely many times in all three directions of space.

Definition 3 *Let* $\mathcal{S}$ *be a point set in* $\mathbb{T}^3$*. By* $DT_{\mathbb{R}}(\mathcal{S})$ *we denote the Delaunay triangulation of* $g(\mathcal{S})$ *in* $\mathbb{R}^3$*. We define the Delaunay triangulation of* $\mathcal{S}$ *in* $\mathbb{T}^3$ *as follows:* $DT_{\mathbb{T}}(\mathcal{S}) := g^{-1}(DT_{\mathbb{R}}(\mathcal{S}))$.

*This work was partially supported by the ANR *(Agence Nationale de la Recherche)* under the "Triangles" project of the *Programme blanc* (No BLAN07-2_194137)
http://www-sop.inria.fr/geometrica/collaborations/triangles/

†INRIA Sophia-Antipolis (Manuel.Caroli@sophia.inria.fr)

‡Nico Kruithof worked on this during his stay at INRIA Sophia-Antipolis (Kruithof@jive.nl)

§INRIA Sophia-Antipolis (Monique.Teillaud@sophia.inria.fr)

In further discussions we will need the following definition:

Definition 4 (domain) *The* domain (i,j,k) *is defined as* $(i,j,k)+[0,1)^3$. *It is a cube into which* g *maps each point of* $\mathbb{T}^3$ exactly *once.*

Our idea is to reuse an algorithm for $\mathbb{R}^3$ to compute the triangulation in $\mathbb{T}^3$. Since g maps the points from $\mathbb{T}^3$ to infinitely many points in $\mathbb{R}^3$ we have to restrict on finitely many domains, when actually computing $DT_{\mathbb{T}}$.

As mentioned in Section 2, a triangulation is defined as a simplicial complex. It might happen that $DT_{\mathbb{T}}(\mathcal{S})$ is not a simplicial complex and thus not a triangulation. As an example, consider $DT_{\mathbb{T}}(\{q\})$. It consists of 7 edges, 12 facets and 6 cells (see Figure 1 for a 2D illustration). Thus it is not a simplicial complex because the vertices of all edges and facets are equal.

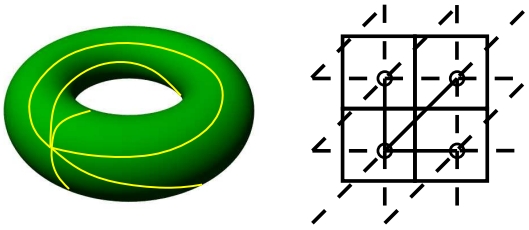


Figure 1: Left: $DT_{\mathbb{T}}(\{q\})$. Right: 4 periodic copies are shown.

4 Coverings

In this section we describe how we overcome the problem that $DT_{\mathbb{T}}$ might not be a simplicial complex. The idea is to use coverings.

Definition 5 (Covering) *A* covering space *of a topological space* $\mathcal{X}$ *is a space* $\mathcal{C}$ *together with a continuous surjective map* $f:\mathcal{C}\to\mathcal{X}$ *that fulfills the following condition: Every point of* $\mathcal{X}$ *has an open neighborhood* $\mathcal{V}$ *such that* $f^{-1}(\mathcal{V})$ *is a disjoint union of open sets in* $\mathcal{C}$.

Note that together with g defined in Section 3, $\mathbb{R}^3$ is a covering space of $\mathbb{T}^3$.

Definition 6 (k-sheeted covering) *The cardinality of* $f^{-1}(x)$ *is locally constant over* $\mathcal{X}$. *If* $\mathcal{X}$ *is connected, this cardinality is constant and it is called the* number of sheets *of the covering.*

For more background in topology cf. [6, 13].

Since $\mathbb{T}^3$ is connected, we can define a k-sheeted covering as follows. We use the notation $\mathbb{T}^3_n$ to denote an n^3-sheeted covering of $\mathbb{T}^3$ induced by the map $h_n:\mathbb{T}^3\to\mathbb{T}^3_n$ defined as:

$$h_n(q) := \{(u,v,w)+(i,j,k) \mid i,j,k\in[0,n)\cap\mathbb{Z}\}$$

In $\mathbb{T}^3_n$ we have a coordinate system with coordinates in $[0,n)^3$.

Proposition 1 *$h_2(DT_{\mathbb{T}}(\mathcal{S}))$ is always a simplicial complex.*

Proof. It is easy to see that it is enough to prove that there are no self-edges in $\mathbb{T}^3_2$. A self-edge in $\mathbb{T}^3_2$ has length ≥ 2, so there are no self-edges if the radius of the largest empty ball is <1. The largest empty ball we can place in $DT_{\mathbb{T}}(\{q\})$ has radius $\frac{\sqrt{3}}{2}$, which is <1. Adding further points cannot increase the radius of the largest empty ball, so self-edges can never occur. □

Since computing in $\mathbb{T}^3_2$ is quite expensive, we want to have a simple criterion telling whether it is possible to compute in $\mathbb{T}^3_1$.

Proposition 2 *Let* $\mathcal{S}\subseteq\mathcal{T}$ *be two point sets in* $\mathbb{T}^3$. *If the longest edge length of* $DT_{\mathbb{R}}(\mathcal{S})$ *is* $<\sqrt{\frac{2}{3}}$, *then* $DT_{\mathbb{T}}(\mathcal{T})$ *is a simplicial complex in* $\mathbb{T}^3_1$.

Proof. If the largest empty ball has radius $<\frac{1}{2}$, then $DT_{\mathbb{T}}(\mathcal{S})$ is a simplicial complex in $\mathbb{T}^3_1$. We assume the existence of an empty ball of radius $\frac{1}{2}$. The longest edge of this triangulation has a length of at least $\sqrt{\frac{2}{3}}$ (regular tetrahedron with circumscribing ball of radius $\frac{1}{2}$). If all the edges of the triangulation are shorter than $\sqrt{\frac{2}{3}}$, then this is a contradiction to the assumption, so then there are no empty balls of radius $\geq\frac{1}{2}$. This means that there are no self-edges and thus $DT_{\mathbb{T}}(\mathcal{S})$ is a simplicial complex in $\mathbb{T}^3_1$. Since adding further points cannot increase the largest empty ball size, even $DT_{\mathbb{T}}(\mathcal{T})$ is a simplicial complex in $\mathbb{T}^3_1$. □

5 Implementation

In this section we describe solutions to several problems we encountered and accelerations we implemented. The algorithm used in the CGAL 3D triangulation package is incremental. For each inserted point p, it first performs a point location, then it marks all cells in conflict with p[1]. At last it updates the data structure by removing all cells in conflict and filling the hole with new cells incident to p.

[1] i.e. cells whose circumscribing ball contains p

5.1 Offsets

In $\mathbb{R}^3$ a tetrahedron is uniquely defined by four vertices. In $g(\mathbb{T}^3)$ four vertices can define several tetrahedra because each vertex corresponds to infinitely many points (cf. Fig. 2 for an illustration of the 2D case). However, in the case of Delaunay triangulation, a tetrahedron can have vertices in different domains only if the boundaries of these domains meet in at least one corner. To specify the desired tetrahedron

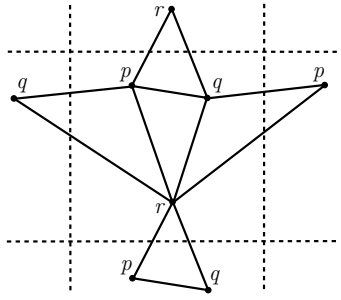


Figure 2: The triangle *pqr* is not uniquely defined in $\mathbb{T}^2$.

we store with each vertex an offset $\in \mathbb{Z}^3$, telling the domain in which it has to be found. The offset is chosen relative to the other offsets in the cell. As vertices have to lie in neighboring domains, we are free to choose the offsets such that we only need to store offsets $\in \{0,1\}^3$ within one tetrahedron. This is useful to accelerate the computation and to save memory.

5.2 2-cycles

If the triangulation contains cycles of length 2, then the union of the closure of all cells in conflict might not be homeomorphic to a ball, and the updating step of the algorithm described in the beginning of this section does not work. More details will be given in a forthcoming publication.

Proposition 3 *Let $\mathcal{S}$ be a point set in $\mathbb{T}^3$. $h_3(DT_{\mathbb{T}}(\mathcal{S}))$ does not contain cycles of length 2.*

Proof. To be sure that there are no cycles of length ≤ 2 in $\mathbb{T}^3_3$, the radius of the largest empty ball has to be $< \frac{3}{4}$. However, the largest empty ball we can place into the point grid generated by one point in $\mathbb{T}^3$ has a radius of $\frac{\sqrt{3}}{2} \approx 0.866 > \frac{3}{4}$.

Since it is not enough to use the size of the largest empty balls, we argue using their position in space: They are centered at $q+\{0,1,2\}^3$ ($q \in [0,1)^3$). Therefore the distance between the centers of two neighboring balls is at least 1 (axis-aligned direction). This implies that it is 2 if we walk along the torus in the other direction because the length of the shortest loop in $\mathbb{T}^3_3$ is 3. Since the radius computed above is clearly < 1 the balls can never overlap on *both* sides and thus a 2-cycle cannot occur. If we consider non-axis-aligned directions it is even easier to argue, because in this case the length of a cycle is larger and the ball radius argument alone is sufficient (length of a cycle $\geq \sqrt{18}$, 4·ball radius $= \sqrt{12}$, $\sqrt{12} < \sqrt{18}$). □

We can reuse the proof of Proposition 2 with the threshold $\frac{1}{2} \cdot \sqrt{\frac{2}{3}} = \frac{1}{\sqrt{6}}$ to prove that there are no more 2-cycles in $\mathbb{T}^3_1$, if every edge is shorter than $\frac{1}{\sqrt{6}}$. Our algorithm works as follows:

- Compute initially in $\mathbb{T}^3_3$.
- Maintain a data structure $\mathcal{D}$ storing pointers to all edges that are longer than $\frac{1}{\sqrt{6}}$.
- Once $\mathcal{D}$ is empty: switch back to $\mathbb{T}^3_1$.

It remains to remark that using this approach does not mean that when computing in $\mathbb{T}^3_1$ all the edges are shorter than $\frac{1}{\sqrt{6}}$ but only that there are no 2-cycles and there cannot occur any by only adding further points.

Note that once we compute in $\mathbb{T}^3_1$, the algorithm does not keep track of edge lengths anymore and thus imposes no more overhead.

This approach also handles vertex removal, the complete description will be given in a full paper.

5.3 Dummy grid points

If the point set for which the triangulation should be computed does not contain 2-cycles in $\mathbb{T}^3_1$, then the usage of $\mathbb{T}^3_3$ and the keeping track of edge lengths takes unreasonably much time. For this case we provide a more efficient possibility. The idea is to precompute a triangulation of a dummy point grid such that we only need to compute in $\mathbb{T}^3_1$ to add further points. Once all the points from the given point set are added, we remove the dummy points from the triangulation. If the number of dummy points is very small compared to the number of all points, the overhead of removing them is negligible.

To implement this idea, we first need to have a dummy point set that is small and easy to precompute. It turns out to be enough to add 27 points in a regular grid, e.g. $\left\{0, \frac{1}{3}, \frac{2}{3}\right\}^3$. The Delaunay triangulation of this point set in $\mathbb{T}^3_1$ is topologically equivalent to the Delaunay triangulation of one vertex in $\mathbb{T}^3_3$. Therefore the proof of Proposition 3 can be reused.

Note that usually, it is not possible to decide whether a triangulation contains 2-cycles beforehand. However, our algorithm works in any case by resorting to $\mathbb{T}^3_3$ during the dummy point removal, if necessary.

5.4 Spatial sort in $\mathbb{T}^3$

To accelerate the point location while inserting point sets, the points are sorted beforehand with the prop-

erty, that two consecutively inserted points are spatially close [5]. Then a cell incident to the vertex added in the step before, can be used as a starting cell for the point location of the next point. This makes the point location very fast.

If we compute in $\mathbb{T}^3$, we encounter the following problem: Cells cut by domain boundaries are always stored such that they are cut by the boundary with larger coordinates. E.g. a cell cut by the horizontal boundary plane is stored to be cut by $z = 1$ and not $z = 0$. So if we add for example a point with a small z coordinate ($z \approx 0$), it might happen that the cell that contains it is stored on the upper boundary of the domain ($z \approx 1$). If the point location starts in a cell close to the point ($z \approx 0$), we will have to traverse the whole domain for the point location.

The solution is as simple as effective: The point location function gets an offset to know in which domain it should operate. So if one of the three coordinates of the point to locate is $< \frac{1}{2}$, we diminish the respective offset entry by 1. In the above example case we would diminish the z-value of the offset by one because the z-coordinate of the point to locate is $< \frac{1}{2}$.

6 Benchmarks

We first compare the computations of Delaunay triangulation in $\mathbb{R}^3$ and $\mathbb{T}^3$. We clearly see that triangulations of small point sets are comparatively very expensive in $\mathbb{T}^3$ because of the overhead due to computing in $\mathbb{T}^3_3$. For large point sets this amortizes and we finally get a factor of about 2. This is not completely satisfactory and we are currently working on further improvements.

no. of points	$\mathbb{R}^3$	$\mathbb{T}^3$
1000	0.0240	2.65
10000	0.244	4.46
100000	2.46	8.35
1000000	25.1	52.3

Using grid points (cf. Section 5.3) improves computing Delaunay triangulations in $\mathbb{T}^3$ especially for smaller point sets:

no. of points	w/o grid	w/ grid
1000	2.65	0.0593
10000	4.46	0.498
100000	8.35	4.72
1000000	52.3	49.6

All benchmarks have been run on an Intel Pentium 4 CPU clocked at 3.6 GHz. The used operating system is Linux Fedora Core 5 and gcc version 4.1.1 with the optimization option `-O2`. The given results are obtained by using `CGAL::Timer` and computing the average of the run time of three runs rounded to three significant digits. All computations have been performed on a random point set, uniformly distributed in a half-open unit cube. The results are given in seconds.

Acknowledgments

We want to thank Oswin Aichholzer, Franz Aurenhammer, Thomas Hackl, Bernhard Kornberger, and Birgit Vogtenhuber from TU Graz for fruitful discussions, in the framework of the *Partenariat Hubert Curien (PHC)*, "Geometric Concepts and CGAL"[2].

References

[1] J.-D. Boissonnat and M. Yvinec. *Algorithmic Geometry*. Cambridge University Press, UK, 1998. Translated by Hervé Brönnimann.

[2] M. Caroli, N. Kruithof, and M. Teillaud. Decoupling the CGAL 3d triangulations from the underlying space. In *Proc. Workshop on Algorithm Engineering and Experiments (ALENEX)*, 2008.

[3] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

[4] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, Germany, 2nd edition, 2000.

[5] C. Delage. Spatial sorting. In CGAL Editorial Board, editor, *CGAL User and Reference Manual*. 3.3 edition, 2007.

[6] A. Hatcher. *Algebraic topology*. Cambridge University Press, 2002.

[7] M. Held. Vroni: An engineering approach to the reliable and efficient computation of Voronoi diagrams of points and line segments. *Comput. Geom. Theory Appl.*, 18:95–123, 2001.

[8] S. Pion and M. Teillaud. 3d triangulations. In CGAL Editorial Board, editor, *CGAL User and Reference Manual*. 3.3 edition, 2007.

[9] Qhull. http://www.qhull.org.

[10] V. Robins. Betti number signatures of homogeneous Poisson point processes *Physical Review E 74 (2006) 061107*.

[11] G. Rote and G. Vegter. Computational topology: An introduction. In J.-D. Boissonnat and M. Teillaud, editors, *Effective Computational Geometry for Curves and Surfaces*. Springer-Verlag, Mathematics and Visualization, 2006.

[12] J. R. Shewchuk. Triangle: Engineering a 2d quality mesh generator and Delaunay triangulator. In *First Workshop on Applied Computational Geometry*. Association for Computing Machinery, May 1996.

[13] W. P. Thurston. *Three-dimensional geometry and topology*. Princeton University Press, 1997.

[14] A. Zomorodian. *Topology for Computing*. Cambridge University Press, Cambridge, 2005.

[2] `http://www-sop.inria.fr/geometrica/collaborations/Amadeus`

Realizability of Solids from Three Silhouettes

Tomohiro Ohgami * Kokichi Sugihara †

Abstract

Given three arbitrary silhouettes whose viewpoints are unknown, we are interested in judging whether there exists a solid that has these three silhouettes and in constructing one if it exists. If we know the viewpoint for each silhouette, we can construct a solid uniquely by the volume intersection method and check whether this solid has given silhouettes. In this paper, we propose an algorithm for searching for the solid which has given silhouettes when their viewpoints are unknown.

1 Introduction

Understanding the 3D shape from various image features is a fundamental and important problem in computer vision. Some images are often given as silhouettes or contours. Silhouette-based techniques do not require finding correspondences between images. Some approaches to reconstructing shapes from silhouettes have been proposed [1, 2, 3, 4, 5]. For instance, there is the volumetric approach that reconstructs an object as the intersection of the cones obtained by back-projecting silhouettes from the corresponding viewpoint as shown in Figure 1. The resulting solid approximates the object which has all silhouettes closely. This simple technique is called volume intersection [6, 7]. Volume intersection technique requires the positions of silhouettes and viewpoints, but this information is not often available. Bottino and Laurentini study the problem of understanding 3D shape from silhouettes when the relative positions of the viewpoints are unknown [2].

In this paper, we study the problem of constructing an object from three silhouettes that are given arbitrarily. In our problem, the relative positions of viewpoints are unknown. We assume that all viewing directions are parallel to the horizontal plane. The object which is constructed by volume intersection has three degrees of freedom under this assumption. Those are two angles and one horizontal displacement. This fact is mentioned closely in Section 2. First, we propose an algorithm of calculating a feasible range of the horizontal displacement when two angles are fixed. Next, we propose a heuristic search algorithm of finding two angles which realize the given silhouettes.

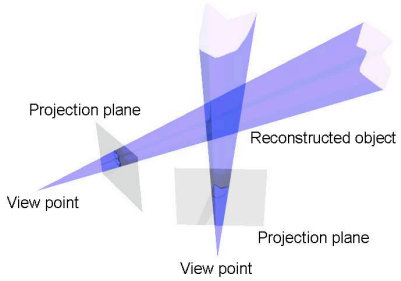


Figure 1: Volume intersection.

2 Preliminaries

The perspective projection and the orthographic projection are used in computer graphics. In this paper, we consider the orthographic projection. That is, the view direction and the corresponding projection plane are perpendicular to each other and the view point is located at infinity. We will restrict ourselves to the case that all viewing directions are horizontal. Clearly, all silhouettes must have the same height and all cylinders obtained by back-projection must be supported by a common horizontal plane. In this setting, the object that is constructed by volume intersection has three degrees of freedom.

Now, suppose that three silhouettes are given. To construct the object by volume intersection, we have to determine three viewing directions and three projection planes corresponding to the silhouettes. Thus, the location and the shape of the constructed object have six degrees of freedom, but we are interested in the shape of the object and hence we consider only three degrees of freedom. Let θ be the angle between the viewing directions of the first silhouette and the second silhouette. Let ϕ be the angle between the viewing directions of the second silhouette and the third silhouette. Let d be the horizontal displacement of the third silhouette on the corresponding projection plane. These notations are used throughout the paper. To find feasible solutions, we must search the three-dimensional space (θ, ϕ, d). In the next section, we fix the two parameters θ and ϕ and we consider a

*Department of Mathematical Informatics, University of Tokyo, tomohiro_oogami@mist.i.u-tokyo.ac.jp
†Department of Mathematical Informatics, University of Tokyo, sugihara@mist.i.u-tokyo.ac.jp

method for determining whether the feasible solutions exist.

3 Feasibility check for the fixed angles

In Section 2, we saw that three degrees of freedom are needed in order to determine the shape of the constructed object. In this section, when two parameters θ and ϕ are fixed, we propose an algorithm for calculating the feasible range of the other parameter d.

First, we place and fix the silhouette on the projection plane coordinate system corresponding to each silhouette. For an arbitrarily chosen k, we consider the horizontal plane $y = k$, cut the silhouette by this plane, we restrict our consideration to this plane. Then, the silhouette consists of line segments. We represent it. A silhouette is represented by the end points of all line segments as follows;

$$S(k) = \bigcup_{i=1}^{p(k)} [x_{2i-1}^{(k)}, x_{2i}^{(k)}], \quad 0 \le k \le y_{\max}, \tag{1}$$

where $S(k)$ is the set of the closed intervals at $y = k$,

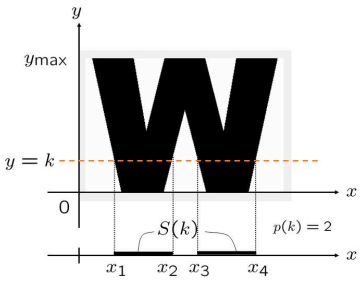


Figure 2: Silhouette cut by a horizontal plane.

and $p(k)$ is the number of connected line segments at $y = k$ as shown in Figure 2. Since we have three silhouettes, we get three sets of line segments at $y = k$ as follows:

$$S_1(k) = \bigcup_{i=1}^{p(k)} [a_{2i-1}^{(k)}, a_{2i}^{(k)}], \tag{2}$$

$$S_2(k) = \bigcup_{i=1}^{q(k)} [b_{2i-1}^{(k)}, b_{2i}^{(k)}], \tag{3}$$

$$S_3(k) = \bigcup_{i=1}^{r(k)} [c_{2i-1}^{(k)}, c_{2i}^{(k)}], \tag{4}$$

$$0 \le k \le y_{\max},$$

where $a_i^{(k)}$, $b_i^{(k)}$ and $c_i^{(k)}$ are the end points of line segments and $p(k)$, $q(k)$ and $r(k)$ are the numbers of connected line segments, respectively. We call $S_1(k)$, $S_2(k)$, $S_3(k)$ the "one dimensional silhouettes" (or just "silhouettes" if there is no ambiguity).

We consider the shape of the 2D object obtained by volume intersection in the cutting plane at $y = k$. We can rewrite the third silhouette by using d as follows:

$$S_3(k, d) = \bigcup_{i=1}^{r(k)} [c_{2i-1}^{(k)} + d, c_{2i}^{(k)} + d], \tag{5}$$

where d is a parameter corresponding to the degree of freedom in the displacement of the third silhouette on the projection plane. An example shape of the 2D object obtained by volume intersection is shown by the shaded areas in Figure 3. Recall that θ is the angle between the viewing directions of the first silhouette and the second silhouette, and ϕ is the angle between the viewing directions of the second silhouette and the third silhouette.

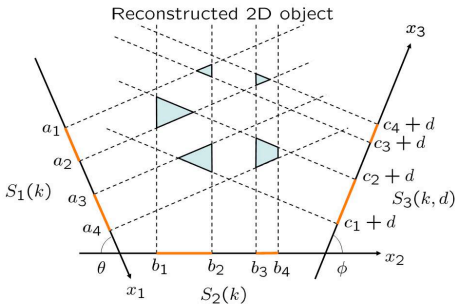


Figure 3: Volume intersection on the plane $y = k$. $p(k) = q(k) = r(k) = 2$.

Once θ , ϕ and d are fixed, we can check whether the set of (θ, ϕ, d) is feasible or not easily by the volume intersection method. Suppose that θ and ϕ are fixed, but d is free. We consider a method for calculating the feasible range of d. By volume intersection, we obtain $p(k)q(k)$ parallelograms from the first silhouette and the second silhouette. Let us number the vertices of those parallelograms from 0 to $4p(k)q(k) - 1$, and let v_i be the value on x_3-axis obtained by projecting the i-th vertex to x_3-axis. Suppose that we displace the third silhouette on the third projection plane by continuously changing the value of d. The feasibility status as to whether all the silhouettes are realizable can change only when one of the end points of the third silhouette hits one of the projected vertices of the parallelograms. Hence, we just have to check at all d that satisfy following equation:

$$c_i + d = v_j, \quad 1 \le i \le 2r(k), \quad 0 \le j \le 4p(k)q(k) - 1. \tag{6}$$

Let $D(k)$ be the range of feasible d at $y = k$. Similar ranges of d can be computed at all k. Let us define

$$D = \bigcap_{0 \le k \le y_{\max}} D(k). \tag{7}$$

If $D \neq \emptyset$, feasible solutions exist for the fixed θ and ϕ. If $D = \emptyset$, feasible solutions do not exist.

4 Heuristic search for parameters (θ, ϕ)

We proposed the algorithm for determining the realizability of solids from silhouettes for fixed θ and ϕ in Section 3. In this section, we propose an algorithm for searching for a solid which has given silhouettes utilizing the algorithm proposed in Section 3.

As we saw in Section 2, there are three degrees of freedom, corresponding to (θ, ϕ, d), when we reconstruct a solid by volume intersection under our assumption. When two of these parameters θ, ϕ are fixed, we can calculate the range of parameter d where a set of parameters (θ, ϕ, d) is feasible. If the feasible range of d is empty, there is no feasible set of parameters at this pair of (θ, ϕ). The problem that needs to be solved next is how to calculate the feasible region of (θ, ϕ) in parameter space. The simplest way for this problem is to search with a fine tooth comb in all (θ, ϕ). Both θ and ϕ are continuous quantity, and hence we need to discretize these value. When we use a large step size for the discretization, we are apt to overlook a tiny feasible region. When we use a small step size, it requires a large amount of time to finish the computation.

We do not want to check all the combination of (θ, ϕ, d). Hence, we consider a heuristic search to find one of feasible solutions. The first good point of this method is that we can expect we do not overlook a tiny feasible region. The second good point of this method is that it takes less time in computation. We start with an initial set (θ, ϕ, d). This set needs not be feasible. We update this set by using the evaluation function which we will propose in the following.

Given silhouettes are represented as follows:

$$S(k) = \bigcup_{i=1}^{p(k)} [x_{2i-1}^{(k)}, x_{2i}^{(k)}]. \tag{8}$$

Now, we define the filled silhouette for this given silhouette by

$$S'(k) = [x_1^{(k)}, x_{2p(k)}^{(k)}]. \tag{9}$$

When we cut a filled silhouette at any $y = k$, we obtain a single line segment. This means that we obtain the filled silhouettes by filling the breach of given silhouettes as in Figure 4. Then, following proposition is satisfied.

Proposition 1 *The feasible region of parameters for filled silhouettes includes the feasible region of parameters for original silhouettes.*

According to Proposition 1, the set of parameters (θ, ϕ, d) we want to obtain should be feasible also for filled silhouettes.

Let us consider the horizontal plane at $y = k$. When two parameters (θ, ϕ) are fixed, we can compute the range of the other parameter d. The range of d for

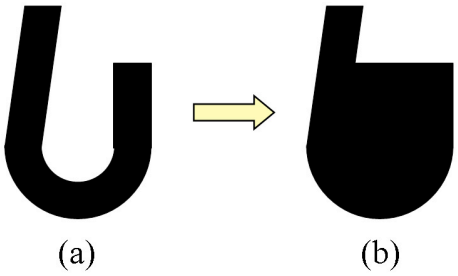


Figure 4: (a) An original silhouette and (b) a filled silhouette.

filled silhouettes consists of one connected component at $y = k$. Let $d_{\min}^{(k)}$ be the minimum value of parameter d at $y = k$, let $d_{\max}^{(k)}$ be the maximum value of parameter d at $y = k$. Let us define

$$d_{\min} = \max_{0 \le k \le y_{\max}} d_{\min}^{(k)}, \tag{10}$$

$$d_{\max} = \min_{0 \le k \le y_{\max}} d_{\max}^{(k)}. \tag{11}$$

The feasible range of d is represented by inequalities

$$d_{\min} \le d \le d_{\max}. \tag{12}$$

We propose an evaluation function as follows:

$$f(\theta, \phi) = d_{\max} - d_{\min}. \tag{13}$$

When we choose a pair of (θ, ϕ) that satisfies $f(\theta, \phi) > 0$, a set of feasible parameters (θ, ϕ, d) exists. When we choose a pair of (θ, ϕ) that satisfies $f(\theta, \phi) < 0$, a set of parameters (θ, ϕ, d) is infeasible for any d. If (θ, ϕ) satisfies $f(\theta, \phi) = 0$, we obtain only one set of feasible parameters (θ, ϕ, d) that satisfies $d = d_{\max} = d_{\min}$.

We calculate the pair (θ^*, ϕ^*) that maximizes $f(\theta, \phi)$. If $f(\theta^*, \phi^*) < 0$, we know that there is no set of feasible parameters. Algorithm 1 calculates the pair (θ^*, ϕ^*) which maximizes $f(\theta, \phi)$ in local area.

There is no guarantee for this algorithm to be able to always find a feasible set of parameters, but our experiments show that the algorithm could find feasible set successfully.

5 Experimental results

In this section, we illustrate an example of our experimental results. Suppose that the three silhouettes are given as shown in Figure 5.

First, we calculated the feasible region by the exhaustive search for θ and ϕ by the step size 1°. This result is as shown in Figure 6. The dark gray areas show the feasible region for both original silhouettes and filled silhouettes, while the light gray areas show the feasible region only for filled silhouettes.

Algorithm 1 Heuristic search

Input: Silhouettes S_1, S_2, S_3,
initial solution $(\theta_0, \phi_0), h_0, h_f$.
Output: (θ^*, ϕ^*).
Procedure:
1: $(\theta, \phi) := (\theta_0, \phi_0), h := h_0$
2: **while** $h > h_f$ **do**
3: Calculate $f(\theta, \phi)$, $f(\theta + h, \phi)$, $f(\theta - h, \phi)$, $f(\theta, \phi + h)$, $f(\theta, \phi - h)$.
4: **if** $f(\theta, \phi)$ is not the largest **then**
5: replace (θ, ϕ) with the largest pair.
6: **else if** $f(\theta, \phi)$ is the largest **then**
7: $h := \frac{h}{2}$.
8: **Output** (θ, ϕ) as (θ^*, ϕ^*)

Figure 5: Given three silhouettes.

Figure 6 shows the trajectory of the search path generated by our heuristic method. We chose 121 initial sets $(\theta_0, \phi_0) = (30i°, 30j°), i, j = 0, 1, \ldots, 10$, and 88 of them arrived in the feasible region for filled silhouettes. For example, when we chose $(\theta_0, \phi_0) = (60°, 60°)$ as the initial set and $h = 0.01$, we obtained the final set $(\theta^*, \phi^*) = (38.6875°, 50.75°)$.

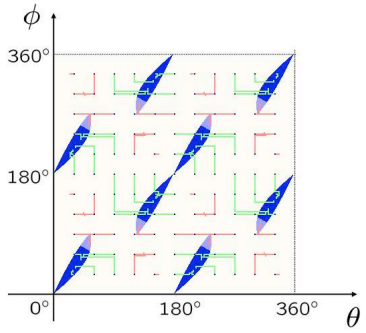


Figure 6: Feasible set of parameters.

In Figure 7, we illustrate the object which has the original silhouettes constructed by these values.

6 Conclusion

We considered the problem of constructing the object which has all aimed silhouettes when three user-established silhouettes are given. We first proposed the algorithm of determining whether the object which have the given silhouettes when the two degrees

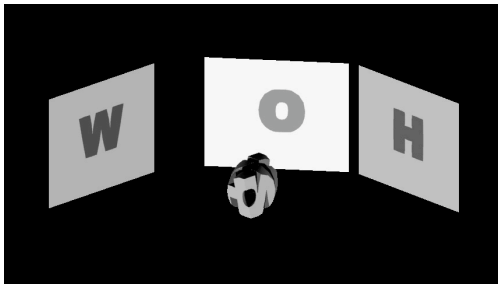


Figure 7: A solid which has given silhouettes.

of freedom are fixed. We next proposed the heuristic search algorithm of finding two angles which realize the given silhouettes. Experiments showed good performance of the proposed algorithm.

Future work includes (1) the construction of sufficient condition for the infeasibility, (2) the improvement of the search method, and (3) extension to few or more silhouettes.

References

[1] A. Bottino, L. Cavallero and A. Laurentini, "Interactive reconstruction of 3D objects from silhouettes." *Proc. Ninth Int'l Conf. in Central Europe on Computer Graphics, Visualization, and Computer Vision*, vol.2, pp. 230–236, 2001.

[2] A. Bottino and A. Laurentini, "Introducing a new problem: Shape-from-silhouette when the relative positions of the viewpoints is unknown." *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 25, pp. 1484–1493, 2003.

[3] A. Laurentini, "The visual hull concept for silhouette-based image understanding." *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 16, pp. 150–162, 1994.

[4] A. Laurentini. "How far 3d shapes can be understood from 2d silhouettes." *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 17, pp. 188–195, 1995.

[5] A. Laurentini, "How many 2D silhouettes it takes to reconstruct 3D objects." *Computer Vision and Image Understanding*, vol. 67, pp. 81–87, 1997.

[6] H. Noborio et al., "Construction of the octree approximating three-dimensional objects by using multiple views." *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 10, pp. 769–782, 1988.

[7] M. Potemesil, "Generating octree models of 3D objects from their silhouettes in a sequence of images." *Computer Vision, Graphics and Image Processing*, vol. 40, pp. 1–29, 1987.

Good Visibility Maps on Polyhedral Terrains

Narcís Coll* Narcís Madern* J. Antoni Sellarès*

Abstract

Let V be a set of view points on or over a polyhedral terrain $\mathcal{T}$. The good visibility of a point q on $\mathcal{T}$ determined by V describes the relationship between q and the distribution of the points in V from which q is visible. A point q on $\mathcal{T}$ is t-well visible relatively to V if and only if every closed halfspace defined by a vertical plane through q contains at least t points of V that are visible from q. The greater the number t the better the visibility of q. The good visibility depth of q is the maximum t such that q is t-well visible relatively to V. The good visibility map of $\mathcal{T}$ determined by V is the subdivision of the terrain $\mathcal{T}$ in good visibility regions where all points have the same fixed good visibility depth. In this paper we present algorithms for computing and efficiently drawing, using graphics hardware capabilities, good visibility maps on polyhedral terrains.

1 Introduction

Good visibility (or illumination) in the plane, introduced by Canales et al. [6, 3, 2, 1], combines two well studied concepts: visibility with obstacles and location depth. Given a set of points P and a set of segment obstacles S, a point q is t-well visible relatively to P and S if and only if every closed halfplane defined by a line through q contains at least t points of P visible from q. So the good visibility of a point q describes the relationship between q and the distribution of the points in P from which q is visible taking into account the effect of the segments of S. The good visibility map determined by P and S is the subdivision of the plane in regions whose points have the same good visibility relative to P and S. Coll et al. [7, 8] present algorithms for computing and efficiently drawing, using graphics hardware capabilities, the good visibility map determined by P and S. The drawing algorithm is also extended to the case of restricted view points, for example visible within an angular region or/and with limited range.

Let $\mathcal{T}$ be a polyhedral terrain, a polyhedral surface that is intersected by any vertical line at most once, represented as a mesh consisting of m triangular faces. The domain $\mathcal{D}$ of $\mathcal{T}$ is the connected region of the xy-plane covered by the triangles obtained vertically projecting the faces of $\mathcal{T}$. A point q on $\mathcal{T}$ is visible from a point v on or above $\mathcal{T}$ if and only if the interior points of the line segment $\overline{vq}$ lie above $\mathcal{T}$. The *visibility map* for v, denoted T_v, is the subdivision of $\mathcal{T}$ into visible and invisible maximal connected components. The orthogonal projection of T_v onto the domain $\mathcal{D}$ defines a planar subdivision D_v of $\mathcal{D}$. The combinatorial complexity of T_v, and consequently of D_v, might be $\Omega(m^2)$. Reif and Sen [14] developed an $O((m+k)\log m \log\log m)$ time algorithm to compute the visibility map T_v, where k is the combinatorial complexity of T_v. Katz et al. [11] presented an output-sensitive algorithm for computing T_v that runs in $O((m\alpha(m)+k)\log m)$ time, where $\alpha(m)$ is the extremely slowly growing inverse of the Ackermann function. Ben-Moshe et al. [5] presented a generic radar-like algorithm for computing an approximation of T_v. Multi-visibility maps related to a set V of view points are obtained combining the visibility maps of the elements in V according to different criteria, typically: union, intersection, counting and overlay [9].

In this paper we extend the good visibility concept to the case of a set V of view points on or over a polyhedral terrain $\mathcal{T}$. The good visibility map of $\mathcal{T}$ determined by V is the subdivision of $\mathcal{T}$ in good visibility regions whose points have the same good visibility. Drawing the good visibility map of $\mathcal{T}$ determined by V helps to visualize the distribution of the points of V relative to the faces of $\mathcal{T}$. We present algorithms for computing and efficiently drawing, using graphics hardware capabilities, the good visibility map of $\mathcal{T}$. We also extend the drawing algorithm to the case of view points visible within an angular region or/and with limited range.

2 Planar depth maps

Let P be a set of n points in the plane. The location depth of an arbitrary point q relative to P, denoted by $ld_P(q)$, is the minimum number of points of P lying in any closed halfplane defined by a line through q. The k-th depth region of P, represented by $dr_P(k)$, is the set of all points q with $ld_P(q) = k$. For $k \geq 1$, the external boundary $dc_P(k)$ of $dr_P(k)$ is the k-th depth contour of P. The depth map of P, denoted $dm(P)$, is the set of all depth regions of P . The complexity

*Institut d'Informàtica i Aplicacions, Universitat de Girona, Spain, {coll,nmadern,sellares}@ima.udg.es. Partially supported by grant TIN2007-67982-C02-02. Narcis Madern is also partially supported by grant BES-2005-9541.

of $dm(P)$ is $O(n^2)$. This bound is tight, for example, when all points of P are in convex position. We denote $dm_r(P)$ the restriction of $dm(P)$ to a planar region r.

2.1 Computing depth contours

Miller *et al* [13] present an algorithm for computing the depth contours for a set P of points n in the plane that makes an extensive use of duality. First, the algorithm maps the points of P to their dual arrangement of lines. Then, a topological sweep is applied to find the planar graph of the arrangement and its vertices are labeled with their levels (the number of dual lines above them). Finally, for a given k, $dc_P(k)$ is computed by finding the lower and upper convex hulls of the vertices at depth k. The complexity of this algorithm, that has been shown to be optimal, is $O(n^2)$ in time and space. In [10, 12] an algorithm is presented that draws, using graphics hardware capabilities, an image of the depth contours as a set of colored pixels, where the color of a pixel is its depth value. The algorithm consists of two steps: in the first step, the input point set P is scan-converted to lines in the dual image plane. The algorithm runs on two bounded duals due to the finite size of the dual plane, in order to guarantee that all intersection points of the lines lie in this finite region. Since each dual plane is discrete, it is possible to compute the level of each pixel by drawing the region situated above every dual line of P, incrementing by one the stencil buffer for each region. In the second step, the two images formed by all the dual lines are scanned, and for each pixel on a dual line the corresponding primal line at the appropriate depth is rendered as a colored 3D graphics primitive using the z-buffer.

3 Good visibility maps on polyhedral terrains

Let $\mathcal{T}$ be a polyhedral terrain composed of m triangular faces and V be a set of n view points on or over $\mathcal{T}$. A point q on $\mathcal{T}$ is t-well visible relatively to V if and only if every closed halfspace defined by a vertical plane through q contains at least t points of V visible from q. The good visibility depth of q relative to V, denoted by $gvd_V(q)$, is the maximum t such that q is t-well visible relatively to V. The k-th good visibility region relative to V, denoted $gvr_V(k)$, is the set of all points q on $\mathcal{T}$ such that $gvd_V(q) = k$. Observe that $gvr_V(k)$ can be formed by several connected components of $\mathcal{T}$. The good visibility map of $\mathcal{T}$ relative to V, denoted $gvm(V)$, is the subdivision of $\mathcal{T}$ determined by the set of all k-th good visibility regions $gvr_V(k)$.

Let V_q be the subset of points of V that are visible from a point q on $\mathcal{T}$. We denote q^*, V^* and V_q^* the orthogonal projection of q, V and V_q onto the domain $\mathcal{D}$ of $\mathcal{T}$, respectively. We also denote $gvr_V(k)^*$ the orthogonal projection of the k-th good visibility region $gvr_V(k)$ onto $\mathcal{D}$. Finally, we denote $gvm(V)^*$ the subdivision of $\mathcal{D}$ determined by the set of all regions $gvr_V(k)^*$, and $gvm_r(V)^*$ the restriction of $gvm(V)^*$ to a region r of $\mathcal{D}$.

3.1 Computing good visibility maps

We will first compute $gvm(V)^*$ on the domain $\mathcal{D}$ of $\mathcal{T}$ and next we will lift up $gvm(V)^*$ to $\mathcal{T}$ to obtain $gvm(V)$. Our algorithm to compute $gvm(V)^*$ will be based on the following

Lemma 1 *For any $q \in \mathcal{T}$: $gvd_V(q) = ld_{V_q^*}(q^*)$.*

The algorithm starts computing for each one of the n points $v \in V$ the visible region T_v of $\mathcal{T}$ and its orthogonal projection D_v onto the domain $\mathcal{D}$. This can be done, by using the algorithm of Katz et al. [11], in $O(n((m\alpha(m) + k)\log m))$ time, where $k \in O(m^2)$ is the maximal combinatorial complexity among the n visibility maps T_v. The overall combinatorial complexity of the n visibility maps T_v, and consequently of their n projections D_v, is $O(nm^2)$.

Next, the algorithm computes the overlay $\mathcal{O}$ of the n planar subdivisions D_v. All points in a cell c of $\mathcal{O}$ are seen from exactly the same subset V_c of points of V. Observe that may exist two cells $c \neq c'$ of $\mathcal{O}$ so that $V_c = V_{c'}$, it is to say the points in c and c' are seen from the same subset of points of V. The overlay $\mathcal{O}$ can be computed by using an algorithm for finding segments intersections. Since there exist an optimal algorithm that finds the k intersections between n segments in $O(n \log n + k)$ time and $O(n)$ space [4], the overlay $\mathcal{O}$ can be computed in $O(nm^2 \log nm + k)$ time and $O(nm^2)$ space, with $k \in O(n^2m^4)$.

Finally, for each cell c of $\mathcal{O}$ whose points are seen from the subset V_c of V, Lemma 1 states that $gvm_c(V_c^*)$ can be computed as $dm_c(V_c^*)$, the depth map of the set V_c^* restricted to c. Then, we have:

$$gvm(V)^* = \bigcup_{c \in \mathcal{O}} dm_c(V_c^*)\,.$$

We compute $dm_c(V_c^*)$ by intersecting the cell c with the depth contours determined by $dm(V_c^*)$. Assuming that on average the complexity of the cell c is constant (however, the complexity of a single cell can be superlinear in the worst case) and since $|V_c^*| \in O(n)$, this can be done in $O(n^2)$ time.

So, putting this all together we can conclude with the following

Theorem 2 *We can compute $gvm(V)^*$, and consequently the good visibility map $gvm(V)$, in $O(n^4m^4)$ time.*

4 Drawing good visibility maps

The implemented solution uses GPU capabilities to solve the problem. The algorithm has four steps:

1. Approximating D_v, for every $v \in V$

The first step obtains an approximation of the orthogonal projection D_v of T_v onto $\mathcal{D}$, for every view point $v \in V$.

We associate to each point v of V a bit of the RGBA channel: the first point of V has associated the first RGBA bit, the second point the second bit, etc. Observe that our approach is restricted to a maximum of 32 view points.

First of all, for each point $v \in V$, we paint the projection of the faces of $\mathcal{T}$ on $\mathcal{D}$. For each v we use a pixel shader on the GPU to obtain a texture F_v. The RGBA buffer of each texel of F_v stores the index of its corresponding face f on $\mathcal{T}$. During this process we mark the texels of the back faces of $\mathcal{T}$ when seen from v, so that they are not considered any more during the whole process.

Next, in the CPU, we visit all texels t in F_v in order to generate a texture G_v representing D_v. For each t with associated point q on $\mathcal{T}$ we have to test if $\overline{vq}$ is partially below $\mathcal{T}$, in which case q is not visible from v. This test can be done by using a *line of sight* GPU technique (see [15] for more details) based on the OpenGL *Occlusion Query* extension. In our case, the occlusion query returns the number of pixels of the segment $\overline{vq}$ covered by the triangles of the terrain. If the number of pixels is 0 we assign 1 to the RGBA bit associated to v of the G_v texel t, otherwise we assign 0. Since it is possible that during the testing process, due to rasterization, the face f, where is located q, can be tested erroneously as partially occluded by itself, the segment $\overline{vq}$ is modified before the occlusion query test by clipping it with vertical planes through the edges of f.

2. Computing the overlay $\mathcal{O}$ of all D_v

We represent the overlay $\mathcal{O}$ of the planar subdivisions D_v, $v \in V$, as a texture O that can be computed with the OpenGL *LogicOp* operation using the textures G_v. Since we have used a bit of color for each point of V, we can assure that any combination of colors in a cell c of $\mathcal{O}$ will correspond to a distinct subset of points of V, so we know from which subset V_c of points of V is visible each c, simply by looking into the bits of its color.

3. Computing $dm_c(V_c^*)$ for every cell c of $\mathcal{O}$

By adapting the second part of the algorithm for computing good visibility maps in 2D [7, 8], we compute $dm_c(V_c^*)$ for each distinct subset V_c^*. In this way we obtain $gvm(V)^*$ on $\mathcal{D}$.

4. Drawing $gvm(V)$

Finally we have to map $gvm(V)^*$ to $gvm(V)$ on $\mathcal{T}$. Using graphics hardware this process is a mapping of the texture O on the faces of $\mathcal{T}$.

4.1 Results

We have implemented the proposed method using C++ and OpenGL, and all the tests and images have been carried out on a Intel Pentium at 3GHz with 2GB of RAM and a GeForce 7800 graphics card using a screen resolution of 500x500 pixels.

Figures 1 and 2 show some examples of good visibility maps on a terrain obtained using our implementation. In these figures $\mathcal{T}$ is colored in a grey gradation according to its good visibility depth (black corresponds to level one, but white corresponds to level zero). The light grey spheres represent the view points of V.

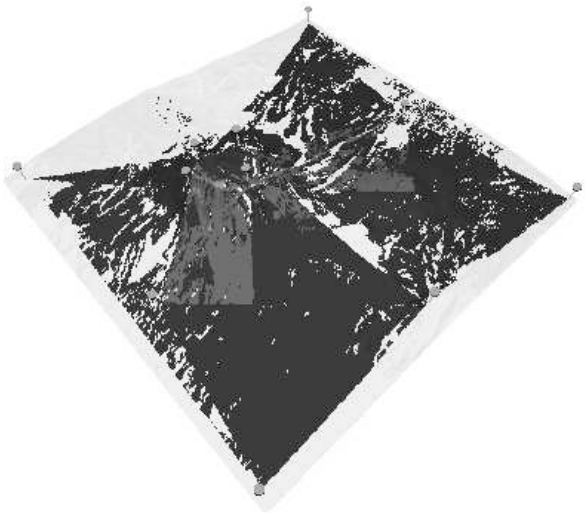


Figure 1: *We can observe $gvm(V)$ from a set of 11 view points on an approximation with 110.000 faces of the Mount Kilimanjaro.*

As an example of running time in the particular cases of Figures 1 and 2, our algorithm takes 38 and 87.5 seconds for the whole process, respectively. An important part of these times is dedicated to the computation of $\mathcal{O}$, 10 and 13.5 seconds respectively.

5 Future work

We are improving some parts of the implemented algorithm to reduce its running time. In particular, we are programming a method that is executed completely inside the GPU for computing $\mathcal{O}$ without using

Figure 2: *The image shows $gvm(V)$ from a set of 14 view points on $\mathcal{T}$, a representation of the Mont Blanc mountain with 40.000 faces.*

occlusion query calls. In this way, the time needed to compute $\mathcal{O}$ will be reduced at least one order of magnitude.

We are also working in the study and implementation of Good Visibility Maps when the view points have limited visibility [1, 2, 7, 8].

Another important future work is the study and implementation of Good Visibility Maps in 3D, where the view points can be placed in any position of the space and obstacles are represented by triangles.

References

[1] M. Abellanas, A. Bajuelos, G. Hernández, and I. Matos. Good illumination with limited visibility. In Wiley-VCH Verlag, editor, *Proc. International Conference of Numerical Analysis and Applied Mathematics*, pages 35–38, 2005.

[2] M. Abellanas, A. Bajuelos, and I. Matos. Good θ-illumination of points. In *Proc. 23rd European Workshop on Computational Geometry*, pages 61–64, 2007.

[3] M. Abellanas, S. Canales, and G. Hernández. Buena iluminación. In *Actas de las IV Jornadas de Matemática Discreta y Algorítmica*, pages 239–246, 2004.

[4] I. J Balaban. An optimal algorithm for finding segments intersections. *Proc. 11th Sympos. on Comput. Geom.* pages 211–219, 1995.

[5] B. Ben-Moshe, P. Carmi and M. J. Katz. Approximating the Visible Region of a Point on a Terrain. *GeoInformatica*, to appear 2008.

[6] S. Canales. *Métodos heurísticos en problemas geométricos, Visibilidad, iluminación y vigilancia.* PhD thesis, Universidad Politécnica de Madrid, 2004.

[7] N. Coll, M. Fort, N. Madern, J.A. Sellarès. GPU-based Good Illumination Maps Visualization. *Actas XII Encuentros de Geometría Computacional*, pages 95–102, 2007.

[8] N. Coll, M. Fort, N. Madern and J.A. Sellarès. Good Illumination Maps. *23th European Workshop on Computational Geometry*, pages 65–68, 2007.

[9] N. Coll, M. Fort, N. Madern and J.A. Sellarès. Multi-visibility maps of triangulated terrains. *International Journal of Geographical Information Science*, Vol. 21, No. 10, pages 1115-1134, 2007.

[10] Fischer I. and Gotsman C. Drawing Depth Contours with Graphics Hardware. *Proceedings of Canadian Conf. on Comp. Geometry*, pages 177–180, 2006.

[11] M. J. Katz, M. H. Overmars, and M. Sharir. Efficient hidden surface removal for objects with small union size. *Comput. Geom. Theory Appl.*, 2:223-234, 1992.

[12] Shankar Krishnan, Nabil H. Mustafa, and Suresh Venkatasubramanian. Hardware-assisted computation of depth contours. *Proc. of thirteenth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 558–567, 2002.

[13] K. Miller, S. Ramaswami, P. Rousseeuw, J.A. Sellarès, D. Souvaine, I. Streinu and A. Struyf. Fast implementation of depth contours using topological sweep. *Statistics and Computing*, 13:153–162, 2003.

[14] J. H. Reif and S. Sen. An efficient output-sensitive hidden surface removal algorithm and its parallelization. *Symposium on Computational Geometry*, pages 193-200, 1988.

[15] Brian Salomon, Naga K. Govindaraju, Avneesh Sud, Russell Gayle, Ming C. Lin, and Dinesh Manocha. Accelerating line of sight computation using graphics processing units. In *Proc. of Army Science Conference*, 2004.

Directly Visible Pairs and Illumination by Reflections in Orthogonal Polygons

Mridul Aanjaneya* Arijit Bishnu* Sudebkumar Prasant Pal*

Abstract

We consider *direct* visibility in simple orthogonal polygons and derive tight lower and upper bounds on the number of strictly internal and external visibility edges. We also show a lower bound of $\lceil \frac{n}{2} \rceil - 1$ on the number of *diffuse* reflections required for completely illuminating an orthogonal polygon from an arbitrary point inside it. Further, we derive lower bounds on the combinatorial complexity of the *visibility polygon* of a point source S after $k \geq 1$ specular reflections within special classes of polygons.

1 Introduction

Let P be a simple polygon with n vertices. The *internal (external) visibility graph* [2] of P is a graph with vertex set equal to the vertex set of P, in which two vertices are adjacent if the line segment connecting them does not intersect the exterior (interior) of P. A visibility edge is called *strictly internal* (*strictly external*) [2] if it is not an edge of P and lies completely inside (outside) P. Line segments connecting non-consecutive vertices of the polygon that intersect polygon edges are called *mixed visibility edges* [2]. The edge bc (resp. ab) in Figure 1(i) is a strictly external (resp. internal) visibility edge, and cd is a mixed visibility edge. We focus on a special class of polygons, namely *orthogonal polygons*, in which the internal angle at each vertex is either 90 or 270 degrees.

We consider *direct* visibility and derive a lower bound of $(2n - 6)$ on the sum S of the number of strictly internal and strictly external visibility edges. We also derive an upper bound on S by counting the number of mixed visibility edges. We show these bounds to be tight by constructing two families of orthogonal polygons which achieve these bounds.

Next, we consider visibility with *reflections*. We prove that $\lceil \frac{n}{2} \rceil - 1$ *diffuse* reflections are sometimes necessary for completely illuminating a simple polygon from an arbitrary point inside it by considering a spiral orthogonal polygon. We also derive several lower bounds on the combinatorial complexity of the *visibility polygon* $V_P(S)$ of a point source S after $k \geq 1$ specular reflections within special classes of polygons P. For simple orthogonal polygons, we show that $V_P(S)$ can have $\Omega(n^2)$ *holes* with one reflection. We also show that $V_P(S)$ is simply-connected after at most two reflections in spiral orthogonal polygons, and that $V_P(S)$ can have $\Omega(n)$ holes after $\Theta(n)$ reflections in general spiral polygons.

*Department of Computer Science and Engineering, IIT Kharagpur, {mridul, bishnu, spp}@cse.iitkgp.ernet.in

2 Counting visibility edges

Let $int(P)$ (resp. $ext(P)$, $mix(P)$) denote the number of strictly internal (resp. strictly external, mixed) visibility edges of a polygon P. Determining the sum of the number of strictly internal (i) and strictly external (e) visibility edges, $i + e$, of a simple polygon was posed as an open problem in [1]. This problem was settled by Urrutia in [2]. He also suggested a family of polygons (as shown in Figure 1(ii)) that achieve the bound in Theorem 1.

Theorem 1 (Urrutia [2]) *For any simple polygon P with n vertices, the number of strictly internal and strictly external visibility edges is at least $\lceil \frac{3n-1}{2} \rceil - 4$.*

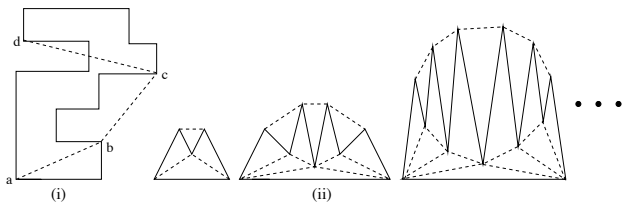


Figure 1: (i) Visibility edges. (ii) A family of simple polygons which achieve the lower bound for $(i + e)$.

2.1 Lower bound on the number of visibility edges

A partitioning of P into convex quadrilaterals is called a *convex quadrilateralization* of P. Only vertices of the polygon P may serve as vertices of the quadrilaterals. Kahn, Klawe and Kleitman [3] proved that every orthogonal polygon P (with or without holes) is convexly quadrilateralizable. Any convex quadrilateralization of P has $\frac{n-2}{2}$ convex quadrilaterals [5]. Adding a diagonal to each quadrilateral gives us a triangulation of P which has $(n - 3)$ edges. If we now flip the diagonal of each quadrilateral, we again get a

triangulation of P with $\frac{n-2}{2}$ distinct edges. Using this fact to count the number of strictly internal visibility edges, we have the following lemma:

Lemma 2 *Any n-sided simple orthogonal polygon P has at least $\frac{3n-8}{2}$ strictly internal visibility edges.*

A vertex v of P is *internal* if it is inside the convex hull of P. We use Lemma 3 proved by Urrutia in [2] to derive a lower bound of $(2n-6)$ on the number of strictly internal and strictly external visibility edges.

Lemma 3 (Urrutia [2]) *If P is a simple polygon with k internal vertices, then there are at least k strictly external visibility edges, i.e., $ext(P) \geq k$.*

Theorem 4 *There are at least $(2n-6)$ strictly internal and strictly external visibility edges in any simple orthogonal polygon P with n vertices.*

Proof. First note that all reflex vertices of P are internal vertices. O' Rourke showed that there are $\frac{n-4}{2}$ reflex vertices in any orthogonal polygon with n vertices [5]. So from Lemma 3, there are at least $\frac{n-4}{2}$ strictly external visibility edges in P. Using this fact and Lemma 2, we get $int(P) + ext(P) \geq 2n-6$. □

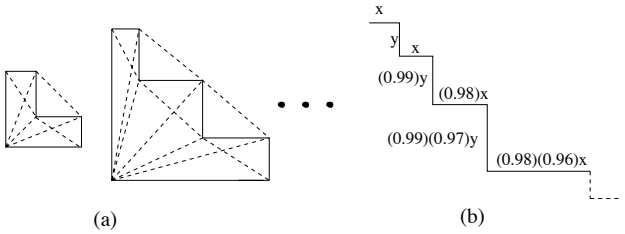


Figure 2: (a) Staircase polygons which achieve the lower bound of $(2n-6)$. (b) Construction scheme for the staircase (x and y are any two positive integers).

We show this bound to be tight by constructing a family of staircase polygons for which $i + e = 2n - 6$ (see Figure 2(a)). A staircase polygon is an isothetic polygon bounded by two monotonically rising (falling) staircases. The staircase is constructed as shown in Figure 2(b).

2.2 Upper bound on the number of visibility edges

In simple polygons, it is easy to see that $int(P) + ext(P) \leq ({}^nC_2 - n)$. This bound is achieved by a convex polygon where the number of mixed visibility edges is zero. However, $mix(P)$ is never zero for orthogonal polygons. We call a horizontal edge of P a *top (bottom)* edge if $int(P)$ is below (above) it. Similarly, we define *left (right)* vertical edges of P. We derive the following upper bound and also show it to be tight by constructing a family of staircase polygons. See Figure 4 for the construction.

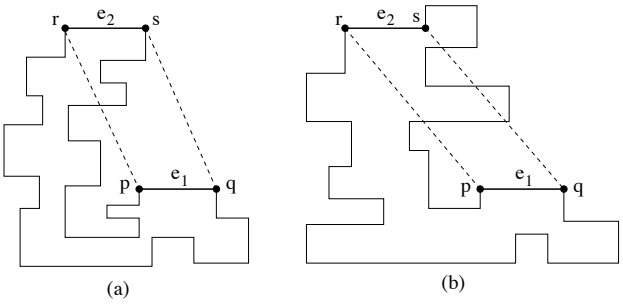


Figure 3: A mixed visibility edge is present between any two top edges (symmetric cases are also possible).

Theorem 5 *There are at most $\frac{n(n-3)}{2} - \sum_{i=1}^{4} \frac{n_i(n_i-1)}{2}$ strictly internal and strictly external visibility edges in any simple orthogonal polygon P with n vertices, where n_1 (resp. n_2, n_3, n_4), is the number of top (resp. bottom, left, right) edges in P.*

Proof. Let e_1 and e_2 be any two top edges in P with e_1 lying below e_2. Let p and q (resp. r and s) be the end-points of e_1 (resp. e_2). See Figure 3. Consider the quadrilateral Q formed by joining p to r and q to s, where traversal of the boundary $bd(P)$ of P in an anticlockwise fashion starting from p gives the sequence of the vertices visited as $p \rightarrow s \rightarrow r \rightarrow q$. Since rs is a top edge, there exists a point x inside Q infinitesimally below rs lying inside P. Similarly, there exists a point y inside Q infinitesimally above pq lying outside P. We conclude that $bd(P)$ intersects the interior of Q. So at least one of the four edges of Q must be a mixed visibility edge. So there is at least one mixed visibility edge corresponding to every pair of top edges. Symmetrically, this claim also holds for every pair of left (resp. right, bottom) edges also. From the above observation we conclude that, $mix(P) \geq \sum_{i=1}^{4} \frac{n_i(n_i-1)}{2}$. Using this observation and the fact that $int(P) + ext(P) + mix(P) = \frac{n(n-1)}{2} - n$, we get $int(P) + ext(P) \leq \frac{n(n-3)}{2} - \sum_{i=1}^{4} \frac{n_i(n_i-1)}{2}$. □

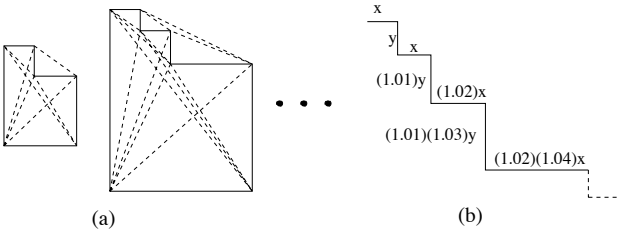


Figure 4: (a) Staircase polygons which achieve the upper bound. (b) Construction scheme for the staircase (x and y are any two positive integers).

3 Visibility with reflections

The problem of visibility when *reflections* from the interior of edges are allowed was first considered in [6, 7, 8]. Two kinds of reflections were defined, *specular*, in which Newton's laws of reflection are obeyed, and *diffuse*, in which light is reflected back in *all* possible directions over a spread of 180 degrees. We denote the *visibility polygon* [4] of a point source S in a simple polygon P by $V_P(S)$.

3.1 Illumination with diffuse reflections

An interesting problem in visibility is to bound the number of diffuse reflections required for completely illuminating a given polygon. The portion m of an edge e that becomes visible at the kth reflection is called a *mirror* at the kth stage. The union of mirrors at the kth stage of reflection, lying on an edge e of P, form connected components called *reflecting segments*. Prasad et al. [8] proved the following lemma:

Lemma 6 ([8]) *Let the edge e of P have a reflecting segment at the lth stage of reflection. Then, the entire edge e is a reflecting segment at the $(l+2)$nd stage.*

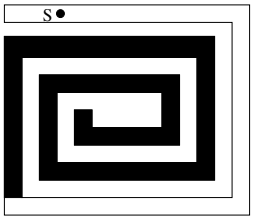


Figure 5: Illumination after two diffuse reflections in spiral polygons. (Illuminated regions are shaded white, non-illuminated regions are shaded black.)

Consider an n-sided spiral orthogonal polygon Q. We claim that $\lceil \frac{n}{2} \rceil - 1$ reflections are necessary for completely illuminating Q from any arbitrary point inside it. Using Lemma 6 one can show that starting from the *arm* containing the source S light successively floods adjoining arms, thereby propagating inside the polygon. So each *arm* of Q requires one diffuse reflection to become illuminated. Hence, Q becomes completely illuminated after $\lceil \frac{n}{2} \rceil - 1$ diffuse reflections. It is well-known that the entire polygon will be illuminated after n diffuse reflections [9]. However, we believe that $\lceil \frac{n}{2} \rceil - 1$ diffuse reflections are also **sufficient** for complete illumination. We have not been able to prove it as yet.

One approach for proving this result would be to study diffuse reflections in the light of the *cooperative guards* problem (see [10]). However, the guards chosen should be *edge guards* instead of *vertex guards* since the latter can see "around the corner", which is not permitted in reflections.

3.2 Combinatorial complexity of visibility polygons with specular reflections

Let P be any n-sided simple polygon. A *blind spot* is a connected component of $P \backslash V_P(S)$. *Holes* are blind spots which do not intersect the boundary of P. Aronov et al. [7] proved that the visibility polygon $V_P(S)$ of a point source S has complexity $\mathcal{O}(n^2)$ with one specular reflection and that this bound is tight. We show that this bound can also be achieved in case of orthogonal polygons, even though the topology of such polygons does not permit the initial angle of incidence of a ray to change significantly. See Figure 6 for an example with $\Omega(n^2)$ holes.

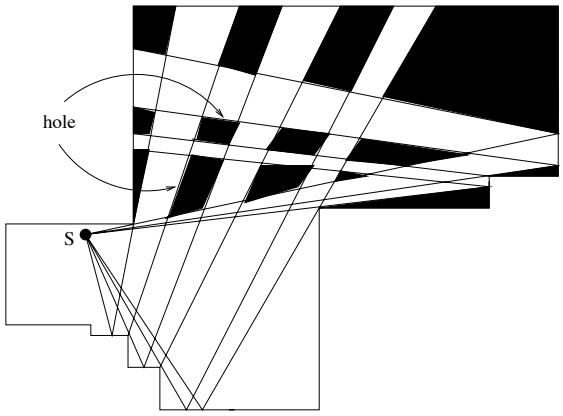


Figure 6: $\Omega(n^2)$ holes with one specular reflection.

Now consider spiral orthogonal polygons. For at most two reflections, we show that $V_P(S)$ is simply-connected in such polygons. For one specular reflection, a *vpath limit* is defined as any 2-link path in P from S to $bd(P)$ that obeys the reflection property but passes through a vertex of P. Blind spots are created by the intersection of vpath limits. Each link of a vpath is given a direction which is same as that of the ray defining it. Suppose two vpath links cross at x. Removal of these two links produces four or more *quadrants*, which are classified as left, right, bottom, or top. If the blind spot lies in the left (right, bottom, top) quadrant with respect to x, locally near x, then x is called a *left* ($right, bottom, top$) vertex. See [7] for details on the above definitions. The following result was proved in [7]:

Lemma 7 ([7]) *A hole is a closed convex polygon with each vertex on two vpath limits; it contains exactly one top and one bottom vertex.*

Lemma 7 can be generalized to multiple specular reflections. Using this result, one can show that at least three sources (real or virtual), which throw light in a shadow, are required for creating a hole at the kth stage of reflection. See Figure 7. There exist only two sources after both one and two specular reflections.

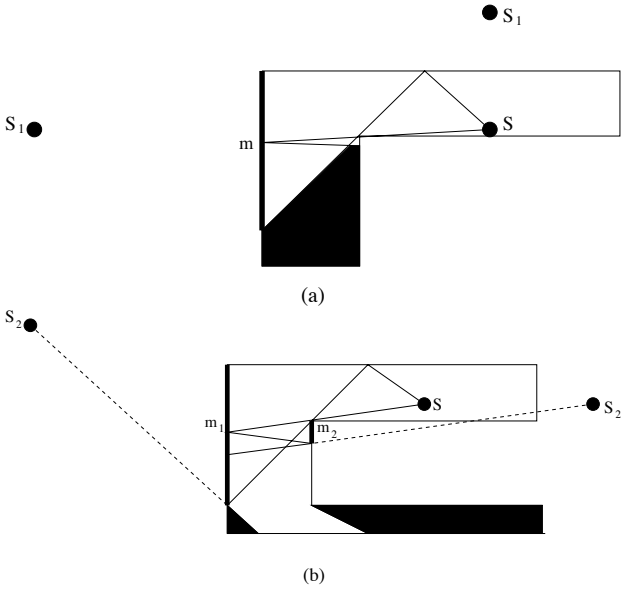


Figure 7: $V_P(S)$ is simply connected after one and two specular reflections.

So $V_P(S)$ remains simply-connected. However, after $k \geq 3$ specular reflections, we observed that the analysis became extremely complicated. We are unaware of the combinatorial complexity of $V_P(S)$ after $k \geq 3$ specular reflections. In case of general spiral polygons however, we observed that $V_P(S)$ is multiply-connected after three specular reflections (see Figure 8). Note that the construction shown in Figure 8 isolates a single beam of light in each *curl* of the spiral polygon. The part of the beam reflected back in the previous curl can be made infinitely thin, so that it has no subsequent effect on the connectedness of $V_P(S)$. We can extend this construction by *curling* the polygon $\Theta(n)$ times to prove the following result:

Theorem 8 *The visibility polygon $V_P(S)$ of a point source S can have $\Omega(n)$ holes in an n-sided spiral polygon P after $\Theta(n)$ specular reflections.*

4 Conclusion

We derived tight lower and upper bounds on the complexity of the internal and external visibility graphs of orthogonal polygons. We considered the problem of illuminating a polygon by multiple diffuse reflections and also studied the complexity of the visibility polygon $V_P(S)$ of a point source S after multiple specular reflections within special classes of polygons. Apart from the several open problems discussed in the paper, we are also studying the complexity of $V_P(S)$ after multiple diffuse reflections. We believe that $V_P(S)$ is always simply-connected in orthogonal polygons. However, we still lack a proof. We also believe that it might be possible to solve the following

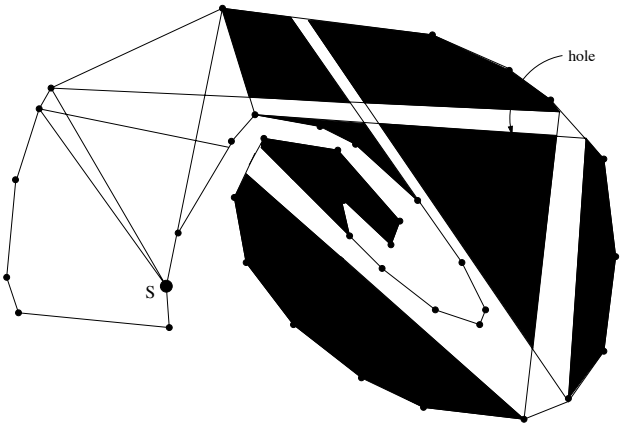


Figure 8: A hole after three specular reflections.

conjecture stated in [8] for orthogonal polygons, and we are currently working towards proving this result.

Conjecture 1 ([8]) *The total complexity of $V_P(S)$ after k diffuse reflections is $\Theta(n^2)$.*

Acknowledgements

The first author would like to thank Monique Teillaud and Olivier Devillers of INRIA, Sophia Antipolis, France for helpful discussions on the topic.

References

[1] J. O' Rourke, *Combinatorics of Visibility and Illumination Problems*, Technical Report, 1996.

[2] J. Urrutia, *On the number of internal and external visibility edges of polygons*, Proc. of CCCG 1997.

[3] J. Kahn, M. Klawe, and D. Kleitman, *Traditional galleries require fewer watchmen*, SIAM J. Alg. Disc. Math. 4(1983), 194-206.

[4] S. K. Ghosh, *Visibility Algorithms in the Plane*, Cambridge University Press, 2007.

[5] J. O' Rourke, *Art Gallery Theorems and Algorithms*, Oxford University Press, 1987.

[6] B. Aronov, A. R. Davis, T. K. Dey, S. P. Pal, and D. C. Prasad, *Visibility with multiple reflections*, Discrete Comput. Geom., 20:61-78, 1998.

[7] B. Aronov, A. R. Davis, T. K. Dey, S. P. Pal, and D. C. Prasad, *Visibility with one reflection*, Discrete Comput. Geom., 19:553-574, 1998.

[8] D. C. Prasad, S. P. Pal, and T. K. Dey, *Visibility with multiple diffuse reflections*, Computational Geometry, 10(3):187-196, 1998.

[9] B. Aronov, A.R. Davis, J. Iacono, and A.S.C. Yu, *The complexity of diffuse reflections in a simple polygon*, Proceedings of LATIN 2006, pp. 93-104.

[10] Pawel Żyliński, *Cooperative Guards in a Fortress Problem*, Balkan Journal of Geometry and its Appl., Vol. 9, No. 2, 2004, pp. 103-119.

Computer Algebra and Computational Geometry

Fabrice Rouillier

SALSA project-team
LIP6
104 avenue du Président Kennedy
75016 Paris, France

e-mail: Fabrice.Rouillier@inria.fr
URL: http://fgbrs.lip6.fr/fabrice/

Abstract

Many algorithms from computer algebra are used in computational geometry such as cylindrical algebraic decomposition for computing the topology of curves. They mostly concern univariate solving (resultants, Sturm sequences, Descartes method, etc.).

In this lecture, the goal is to show that some sophisticated algorithms for exact/certified multivariate solving (solving zero-dimensional systems, study of semi-algebraic sets, etc.) can also be used efficiently in computational geometry in different ways ranging from a straightforward use as plugin for replacing or complementing some numerical methods to dedicated methods for off-line studies to get qualitative information or to prepare a numerical study.

On Planar Visibility Polygon Simplification*

Alireza Zarei† Mohammad Ghodsi†

Abstract

The boundary of a region illuminated by a light source may be composed of many vertices and points. In this paper, we propose a criterion to represent such polygons by a smaller number of vertices and, show how this criterion can be used in offline and streaming models.

1 Introduction

In a planar scene which is composed of a set of polygonal objects in the plane, two points are visible from each other if their connecting segment does not intersect the scene objects. The set of points visible from a point q is called its visibility polygon and is denoted by $VP(q)$. The visibility polygon of a point in a planar domain is always a star-shaped simple polygon. The boundary of a visibility polygon, simply referred to by visibility polygon in the rest of this paper, is composed of many consecutive line segments, some of which may be so far from the observer.

In real applications, an observer usually has a limited vision power, *i.e.*, it can not distinguish small visibility differences at far distances. Moreover, the required space to maintain the exact visibility polygon is too high and it will be impossible to maintain such polygons exactly. On the other hand, the accuracy of the display screens is also limited. That is, to display such a polygon on a display screen, only its approximation is displayed.

In this paper, we consider the problem of simplifying (approximating) the visibility polygon of such observers inside a polygonal domain. This problem is a special case of the well-known line simplification problem for which there are several algorithms. These methods approximate a given path of line segments by another path with smaller number of segments which minimizes the difference between the initial and the simplified paths. This difference, to be formally defined later, is called the *error* of this simplification.

There are two optimization goals in the line simplification algorithms: min-k and min-δ. In the min-k version, there is a given error threshold and we are to use the minimum number of vertices in the simplified path meeting the error threshold. In min-δ, we are allowed to use at most k vertices for some given k in the simplified path and the goal is to minimize the error of the simplification.

Almost all current simplification algorithms solve the line simplification problem under the Hausdorff distance for L_1, L_2 or L_∞ metrics or under the Fréchet distance which are not proper for our purpose of simplifying visibility polygons. In our target applications, the vertices of the path that are closer to the observer are more important than the farther points.

To solve this problem, we define a new approximating error function which considers the distance between the points of the visibility polygon and the observer. We prove that this error function can be computed efficiently and can be used along with current simplification methods without increasing their time or space complexities. Therefore, our target problem can be solved efficiently under min-k or min-δ optimization goals.

We further consider the streaming cases in which the observer is like a radar inside a dynamic environment that circularly sweeps its neighbor and draws its visibility polygon. In such applications, the visible points are given continuously as a stream of input data and we assume that it is impossible to maintain and show all of these points. Therefore, it is necessary to approximate the exact visibility polygon by another polygon of smaller number of vertices.

In this model, regardless of the number of points in the input path, we must simplify the path by at most k points. Also, we must continuously update the simplification as new points are received. For this version of the problem, our proposed method uses $O(\frac{k^2}{\sqrt{\epsilon}})$ additional storage and each point is processed in $O(\frac{k}{\sqrt{\epsilon}\log\epsilon})$ amortized time. Then, the error of the resulting simplification with $2k$ points is not bigger than $(2+\epsilon)$ times the error of the optimal simplification with k points. This method is based on the general algorithm proposed in [1].

There is a similar attempt in rendering based simplification by Buzer [4], however, without considering the observer position. To the best of our knowledge, the results of this paper are the first in this area and there are several interesting open directions in applying and extending this notion.

*This research was in part supported by a grant from IPM. (No. CS1386-2-01)

†Computer Engineering Department - Sharif University of Thechnology, School of Computer Science - Institute for Studies in Theoretical Physics and Mathematics (IPM), zarei@mehr.sharif.edu, ghodsi@sharif.edu

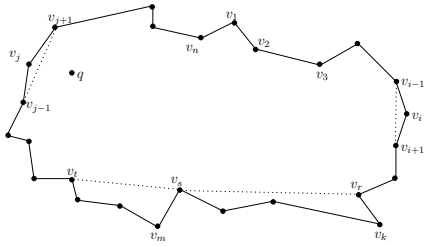


Figure 1: Simplifying $VP(q)$.

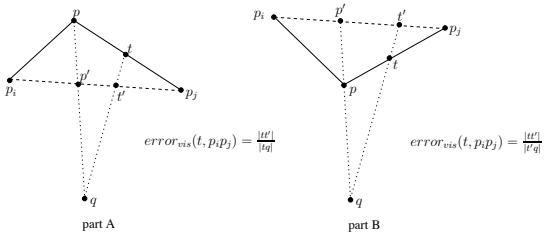


Figure 2: Visibility-dependent simplification error.

2 Visibility-dependent simplification

We focus on the restricted version of the line simplification problem. For this problem, let P be a path defined by a sequence of points $p_0, p_1, p_2, \ldots, p_n$. Any subsequence $Q = q_0, q_1, \ldots, q_l, q_{l+1}$ of P is a l-simplification of P if $q_0 = p_0$ and $q_{l+1} = p_n$. In this simplification, any segment q_iq_{i+1} of Q ($0 \leq i \leq l$) is the corresponding simplification of the subpath $p_s, p_{s+1}, \ldots, p_t$ of P where $q_i = p_s$ and $q_{i+1} = p_t$. In other words, we have replaced the subpath $p_s, p_{s+1}, \ldots, p_t$ of P with segment q_iq_{i+1} in Q.

Therefore, Q is an approximation of P and can be stored using smaller size of memory, however, at the cost of losing the accuracy of P. Assume that *error* is our error function used to compare similarity of Q and P. Using this metric, we denote the error of this approximation by $error(Q)$ and it is defined to be the maximum error of segments $q_iq_{i+1}(0 \leq i \leq l)$ under this metric. The error of a segment q_iq_{i+1} under a metric *error* is denoted by $error(q_iq_{i+1})$ and is defined to be the error of approximating the subpath $p_s, p_{s+1}, \ldots, p_t$ by segment q_iq_{i+1} under this error metric. Usually, the definition of the error metric *error* depends on the application.

Hausdorff error function, $error_h$, is the metric used in almost all simplification algorithms. For a segment q_iq_{i+1} which is the simplification of a subpath $p_s, p_{s+1}, \ldots, p_t$, $error_h(q_iq_{i+1})$ is defined as the maximum euclidian distance of the points $p_s, p_{s+1}, \ldots, p_t$ from segment q_iq_{i+1}.

The Hausdorff error function only depends on the initial and the simplified paths and therefore, is not proper for simplifying visibility polygons in which the position of the observer has an important role. Assume that $P = p_1, p_2, \ldots, p_n, p_1$ of Figure 1 is the visibility polygon of a point observer q. Here, p_j is closer to the observer than p_i which is assumed to be too far from q. If we are to simplify P by removing one point and we have only two choices p_i and p_j, it would be better to remove p_i while if we use Hausdorff error function, p_i will be removed. In order to use current simplification algorithms, we formalize this issue as an error function as follows.

Assume that we are to approximate the path p_ipp_j (See part A of Figure 2), a part of the visibility polygon of the observer q, by segment p_ip_j. From the viewpoint of q, this approximation maps the point p to point p'. Also, other points of segments p_ip and pp_j are mapped to their corresponding points of segments p_ip' and $p'p_j$.

The corresponding visibility-dependent error of this simplification for a point t on the path p_ipp_j is denoted by $error_{vis}(t, p_ip_j)$ and is defined as $\frac{|tt'|}{|tq|}$ where t' is the intersection point of segments tq and p_ip_j. This means that in this simplification and at distance $|tq|$ from the observer we have violated from the initial path by a value of $|tt'|$. This definition is also extended to paths of more internal vertices. The visibility polygon of q is a star-shaped polygon and p lies between p_i and p_j on the boundary of this polygon. Therefore, the supporting line of pq always intersects p_ip_j. If p_i, p_j and q are collinear, p and all other points of the polygon boundary from p_i to p_j must also lie on segment p_ip_j. For such situations, the error of all points of the path from p_i to p_j is zero which corresponds to our definition of the error function.

In some cases like part B of Figure 2, tq does not intersect p_ip_j. For such situations, point t is mapped to point t' which is the intersection point of p_ip_j and the supporting line of tq. In these cases, the visibility-dependent error of point t is defined to be $\frac{|tt'|}{|t'q|}$. Comparing the definition of $error_{vis}$ function for these two cases, when the corresponding values of $|tq|$ in parts A and B of Figure 2 are equal, we assign greater error value to point t in part A. This means that in the same situations we prefer to simplify using the outer diameters of the visibility polygon compare to the internal ones. Another benefit of this definition is that this error function is monotone which will be defined and used in Section 3.

Our visibility-dependent error function associated with a path $p_i, p_{i+1}, \ldots, p_j$ simplified by segment p_ip_j, denoted by $error_{vis}(p_ip_j)$, is defined to be the maximum visibility-dependent error of points of this path.

This definition for visibility-dependent error function strongly relates to the notion of width. The width of a set of points with respect to a given direction $\overrightarrow{d}$ is the minimum distance of two lines being parallel to $\overrightarrow{d}$ that enclose the point set. Let $P_L(i,j)(P_U(i,j))$ be the set of points of subpath $P(i,j) = p_i, p_{i+1}, \ldots, p_j$

that lie in the closed half plane defined by the supporting line of p_ip_j which contains(does not contains) the point observer q. We denote by $w_L(i,j)(w_U(i,j))$ the width of the points of $P_L(i,j)(P_U(i,j))$ with respect to the direction $\overrightarrow{p_ip_j}$. We have,

Lemma 1 *For a subpath $P(i,j) = p_i, p_{i+1}, \ldots, p_j$ of $VP(q)$,*

$$error_{vis}(p_ip_j) = \max\left(\frac{w_U(i,j)}{d(q,p_ip_j)+w_U(i,j)}, \frac{w_L(i,j)}{d(q,p_ip_j)}\right)$$

where $d(q,p_ip_j)$ is the orthogonal distance of point q from the supporting line of p_ip_j.

A direct consequence of this lemma is that the associated error of a segment p_ip_j belongs to a vertex $p_k(i \leq k \leq j)$ which makes computation of this error function straightforward. Using this result we can simply compute the corresponding error of any segment p_ip_j that may appear in simplification during the simplification process by only checking vertices of the subpath $P(i,j)$.

Fortunately, algorithms proposed for both restricted and unrestricted versions of the line simplification problem do not require any special property for the error function and we can plug our error function into. Moreover, this error function can be used under $\min -k$ and $\min -\delta$ optimization goals as well. The only change in these algorithms is to use our error function for a segment p_ip_j when we want to simplify the path $p_i, p_{i+1}, \ldots, p_j$ with this segment.

3 Visibility-dependent simplification in streaming model

In some applications, we can not maintain the whole path because of the limited amount of memory or unnecessity of maintaining these points. For example, consider a radial sweep line which trace the scene around a point observer. In such applications, we want to compute an approximation of the visibility polygon as the visible points are identified by the sweep line.

Formally, the vertices of the visibility polygon are given as a stream of input data and we want to simplify the path. Abam *et.al* proposed a general algorithm that can be used to simplify a path whose vertices are given as a stream of input points[1]. Their algorithm only solves the $\min -\delta$ version of the line simplification problem. Because of the large number of the input vertices, the result of the $\min -k$ version of the line simplification in streaming model, may be too large to store, and therefore, no result exists for.

In order to use this algorithm on a path $P(n) = p_0, p_1, \ldots, p_n$ with an error function *error*, two conditions must be satisfied:

- *error* must be a *c-monotone* error function on the path $P(n)$ for any $n > 0$. This means that for any two segments p_ip_j and p_lp_m such that $i \leq l \leq m \leq j$ and p_i, p_j, p_l and p_m are vertices of the path $P(n)$ we have, $error(p_lp_m) \leq c.error(p_ip_j)$.
- There must be an *e-approximate* error oracle for *error* on the path $P(n)$ to be defined as follows. In streaming models, we may lose some vertices of the subpath $P(i,j)$ between points p_i and p_j. Then, we can not compute the exact value of the error function for this segment and we must approximate this error value. We denote the approximated error value of a segment p_ip_j by $error^*(p_ip_j)$. We call the procedure that computes this approximation as our error *oracle*. An error oracle is e-approximate if for any segment p_ip_j for which the oracle is called by the algorithm we have

$$error(p_ip_j) \leq error^*(p_ip_j) \leq e.error(p_ip_j).$$

Having these two conditions, the algorithm of Abam *et al.* [1] simplifies a streaming path P by a path Q of at most $2k$ internal vertices. The time the algorithm needs to update the simplification upon the arrival of a new point is $O(\log k)$ plus the time spent by the error oracle. Besides the storage needed for the simplification Q, the algorithm needs $O(k)$ storage plus the storage needed by the error oracle. The error of the simplification Q obtained by this algorithm is at most ce times the error of the optimal simplification of P with k points in non-streaming model which we have all points in memory. So, in order to use this algorithm we must show that our visibility-dependent error function, $error_{vis}$, is c-monotone and we must propose an error oracle to approximate the error of any segment p_ip_j for which the oracle is called in this algorithm.

Lemma 2 *Over the visibility polygon of a point observer, the visibility-dependent error function $error_{vis}$ is 2-monotone.*

Proof. Assume that points p_i, p_j, p_l and p_m lie on $VP(q)$ such that $i \leq l \leq m \leq j$ and $error_{vis}(p_lp_m)$ belongs to a point p_k where $l \leq k \leq m$ and p'_k and p''_k are respectively the intersection points of the supporting line of qp_k and segments p_lp_m and p_ip_j. $VP(q)$ is a star-shaped polygon and q is a point in its center. Following the order of points on the boundary of this polygon, the supporting line of segments p_lq, p_mq and p_kq intersect segment p_ip_j and the supporting line of p_kq intersects p_lp_m. There are six permutations for positions of points p_k, p'_k and p''_k on the supporting line of qp_k (shown in Figure 3). For all of these configurations we have

$$error_{vis}(p_ip_j) \geq$$

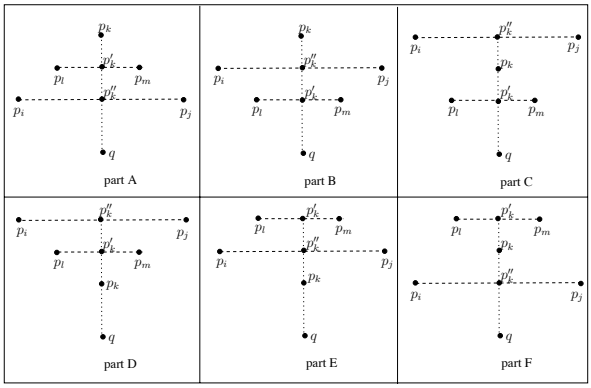


Figure 3: The visibility-dependent error function is 2-monotone.

$\max(error_{vis}(p_l, p_ip_j), error_{vis}(p_m, p_ip_j), error_{vis}(p_k, p_ip_j))$,
and

$$error_{vis}(p'_k, p_ip_j) \leq \max(error_{vis}(p_l, p_ip_j), error_{vis}(p_m, p_ip_j)).$$

Consequently, we have

$$error_{vis}(p_ip_j) \geq \max(error_{vis}(p'_k, p_ip_j), error_{vis}(p_k, p_ip_j)).$$

We prove the lemma for all these configuration by showing that

$$error_{vis}(p_lp_m) = error_{vis}(p_k, p_lp_m)$$
$$\leq 2\max(error_{vis}(p'_k, p_ip_j), error_{vis}(p_k, p_ip_j))$$
$$\leq 2error_{vis}(p_ip_j).$$

The first equality is our assumption that p_k has the maximum error on p_lp_m among all points of path $p_l, p_{l+1}, \ldots, p_m$ and we have already shown the last inequality. Therefore, it is only enough to show the middle inequality. We prove this inequality for the case shown in part A of Figure 3 and skip the other cases. In this configuration we have,

$$error_{vis}(p_k, p_lp_m) = \frac{|p_kp'_k|}{|p_kq|} \leq \frac{|p_kp''_k|}{|p_kq|} = error_{vis}(p_k, p_ip_j)$$
$$\leq 2\max(error_{vis}(p'_k, p_ip_j), error_{vis}(p_k, p_ip_j)).$$

So, we proved that $error_{vis}(p_lp_m) \leq 2error_{vis}(p_ip_j)$. This can be proved for the other cases. Also, it can be shown that this upper bond is tight in cases shown in parts B and E of Figure 3. □

Now, we propose an approximating procedure that approximates $error_{vis}(p_ip_j)$, the error value of any segment p_ip_j for which the simplification algorithm is called.

According to Lemma 1, the approximating oracle can approximate $d(q, p_ip_j)$, $w_L(i,j)$ and $w_U(i,j)$ to find an approximation of $error_{vis}(p_ip_j)$. It is easy to find the exact value of $d(q, p_ip_j)$. We use the method described by Agarwal and Yu [2] to approximate w_L and w_U.

Agarwal and Yu [2] have described a streaming algorithm for maintaining a core-set that can be used to approximate the width of a set of points in any direction. Their algorithm requires $O(\frac{1}{\sqrt{\epsilon}})$ space and $O(\frac{1}{\log \epsilon})$ amortized time per point to maintain a core-set from which the width of the input stream can be computed efficiently. This is done by additionally maintaining the convex hull of the core-set using the data structure by Brodal and Jacob [3]. This data structure uses linear space and can be updated in logarithmic time. Also it supports queries for the extreme point in a given direction in logarithmic time. Using these results, we have an $(1+\epsilon)$-approximate error oracle for $error_{vis}$ and the value of $error_{vis}(p_ip_j)$ can be computed in $O(\frac{1}{\log \epsilon})$ time. Therefore, we can prove the following lemma:

Lemma 3 *There is a $(1+\epsilon)$-approximate error oracle for the visibility-dependent error function on visibility polygon of a point observer that uses $O(\frac{k^2}{\sqrt{\epsilon}})$ storage and has $O(\frac{k}{\sqrt{\epsilon}\log \epsilon})$ amortized update time where k is the number of the internal points of the simplification.*

Combining the result of lemmas 2, 1 and 3 with the algorithm of Abam *et al.* [1] described at the beginning of this section, we have the following result on simplifying the visibility polygon of a point observer based on the visibility-dependent error function:

Theorem 4 *There is a streaming algorithm that maintains a $2k$-simplification for VP(q) under the visibility-dependent error function. This algorithm uses $O(\frac{k^2}{\sqrt{\epsilon}})$ additional storage and each point is processed in $O(\frac{k}{\sqrt{\epsilon}\log \epsilon})$ amortized time and the error of the result simplification is not larger than $(2+\epsilon)$ times the error of the optimal offline k-simplification.*

References

[1] M. A. Abam, M. de Berg, P. Hachenberger, and A. Zarei. Streaming Algorithms for Line Simplification. In *23rd ACM Symp. on Computational Geometry (SoCG)*, pages 175–183, 2007.

[2] P.K. Agarwal, H. Yu. A Space-Optimal Data-Stream Algorithm for Coresets in the Plane. In: *Proc. 23th ACM Symposium on Computational Geometry (SOCG)*, pages 1–10, 2007.

[3] G.S. Brodal and R. Jacob. Dynamic Planar Convex Hull. In *Proc. 43rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 617–626, 2002

[4] L. Buzer. Optimal Simplification of Polygonal Chain for Rendering. In *23rd ACM Symp. on Computational Geometry (SoCG)*, pages 168–174, 2007.

The Kinetic Facility Location Problem*

Bastian Degener †‡ Joachim Gehweiler † Christiane Lammersen †

Abstract

We present a deterministic kinetic data structure for the facility location problem that maintains a subset of the moving points as facilities such that, at any point of time, the sum of the maintenance cost for the facilities and the connection cost for the clients is at most a constant factor larger than the current optimal cost. In our scenario, each point can open a facility and moves continuously along a known trajectory in a d-dimensional Euclidean space where d is a constant.

Our kinetic data structure has a storage requirement of $\mathcal{O}(n(\log^d(n) + \log(nR)))$, where n is the number of points and R is the ratio of the product of the maximum maintenance cost and demand to the product of their corresponding minimum values. In the case that each trajectory can be described by a bounded degree polynomial, the data structure processes $\mathcal{O}(n^2 \log^2(nR))$ events, each requiring only $\mathcal{O}(\log(nR))$ facility changes and $\mathcal{O}(\log^{d+1}(n) \cdot \log(nR))$ time. This results in a total processing time of $\mathcal{O}(n^2 \log^{d+1}(n) \cdot \log^3(nR))$. To the best of our knowledge, this is the first kinetic data structure for the facility location problem.

1 Introduction

The facility location problem (FLP) is a classical combinatorial problem in computer science where we want to find a placement of facilities, like factories or stores of a fast-food chain, such that the combined costs for the maintenance of the facilities and the transportation costs for the customers are minimized.

In this work, we consider a scenario of continuously moving objects in which each object can either be a facility or a client. Applications for this scenario are obviously in sensor networks and mobile ad-hoc networks. In these networks, nodes move continuously and interact with each other. One among many applications is monitoring certain parameters of the sea (e.g. algae population, water pollution) by sensor networks. Imagine those sensors to be a set of micro-submarines to examine parts of the sea. Clearly, the micro-submarines are in motion since they are moved by the currents of the sea. A micro-submarine can be either in sensor mode (client) or in processing mode (facility). In sensor mode, it collects data and sends it to the nearest facility, inducing connection costs depending on the distance. In processing mode, it analyzes the data received and uploads a compressed snapshot via a satellite connection to a research station, which causes maintenance costs. Thus, solving this problem at minimum costs is the FLP.

The kinetic data structure (KDS) framework is well-suited to maintain a combinatorial structure for continuously moving objects and is common in the field of computational geometry [2, 7]. The input is a set of objects and a *flight plan*, i.e., each object moves continuously along a known trajectory. At any point of time, it is possible to change the flight plan by performing a so called *flight plan update*, which means that one object changes its trajectory. The main idea is now, that the continuous motion of the objects is utilized in a way that updates take place only at discrete points of time and can be processed fast. As a result, by maintaining the KDS, a lot of computational effort can be saved compared to handling just a series of instances of the corresponding static problem. This property is called *responsiveness*. Other important properties of a KDS are its *efficiency*, *compactness*, and *locality*.

To guarantee that the required properties of the combinatorial structure are fulfilled at any point of time, a KDS ensures that certain so called *certificates* are always kept. Certificates provide a proof that the combinatorial structure has the desired property. Whenever a certificate fails, we call this an *event* and an update gets necessary.

One challenge to construct a KDS for the FLP is that the underlying combinatorial structure of an exact solution is not stable, i.e., a slight change of the position of a point in an exact solution might require an update on all the points to restore an optimal solution. Therefore, we use a new approach to comply with the condition that an update takes only polylogarithmic time. For a more detailed explanation as well as omitted proofs we refer to [5].

Related work. The KDS framework was introduced by Basch et al. [2]. Only some results are known for problems related to clustering, like the FLP. For in-

*Partially supported by the EU within FP7-ICT-2007-1 under contract no. 215270 (FRONTS), and the DFG-project "Smart Teams" within the SPP 1183 "Organic Computing"

†Heinz Nixdorf Institute and Department of Computer Science, University of Paderborn, {bastian.degener, joge, christiane.lammersen}@upb.de

‡International Graduate School Dynamic Intelligent Systems, University of Paderborn

stance Gao et al. [6] provided a KDS to maintain an expected constant factor approximation for the minimal number of centers to cover all points for a given radius. The centers that they considered are a subset of the moving nodes, like in this paper, whereas Bespamyatnikh et al. [4] studied k-center problems for $k = 1$ in the KDS framework, where the centers are not necessarily located at the moving points. Hershberger [8] proposed a kinetic algorithm for maintaining a covering of the moving points in $\mathbb{R}^d$ by unit boxes. For other work on KDSs, we refer to the survey by Guibas [7].

2 Preliminaries

We define the kinetic FLP as follows. Let $p_1, p_2, \ldots, p_n$ be n independently moving points in $\mathbb{R}^d$, where d is a constant. Let $p_i(t)$ denote the position of p_i at time t and let $\mathcal{P}(t) = \{p_1(t), p_2(t), \ldots, p_n(t)\}$. At any point of time t, the point set $\mathcal{P}(t)$ is divided into two subsets, namely the current set of *facilities* $\mathcal{F}(t)$ and the current set of *clients* $\mathcal{G}(t) = \mathcal{P}(t) \backslash \mathcal{F}(t)$. With each facility $p_i(t) \in \mathcal{F}(t)$, there are non-negative *maintenance costs* f_i associated and each client $p_j(t) \in \mathcal{G}(t)$ has a non-negative *demand* d_j. Note that both the maintenance cost and the demand of a point do not change over time. The problem is now to maintain, at each point of time t, a subset $\mathcal{F}(t) \subseteq \mathcal{P}(t)$, such that

$$cost(\mathcal{F}(t)) := \sum_{p_i(t) \in \mathcal{F}(t)} f_i + \sum_{p_j(t) \in \mathcal{G}(t)} d_j \cdot D(p_j(t), \mathcal{F}(t))$$

is minimized. Here, $D(p_j(t), \mathcal{F}(t))$ is the minimum Euclidean distance from $p_j(t)$ to a facility in $\mathcal{F}(t)$. At first we consider the *uniform* FLP, i.e. $f_i = 1$ for each facility $p_i(t)$ and $d_j = 1$ for each client $p_j(t)$.

Cubes. For a point $p_i(t) \in \mathcal{P}(t)$ and a non-negative value r, we define $\mathcal{C}(p_i(t), r)$ to be the axis-parallel cube whose center is the point $p_i(t)$ and whose side length is $2r$. Given such a cube $\mathcal{C}(p_i(t), r)$, we let $weight(\mathcal{C}(p_i(t), r))$ denote the number of all the points in $\mathcal{P}(t)$ that are located in the cube $\mathcal{C}(p_i(t), r)$.

Radius of a point. For each point $p_i(t) \in \mathcal{P}(t)$, we calculate a special radius $r_i^*(t)$ which is an approximation for the value $r_i(t)$ that is used in [9] and satisfies

$$\sum_{p_j(t) \in \mathcal{P}(t) | D(p_i(t), p_j(t)) \leq r_i(t)} r_i(t) - D(p_i(t), p_j(t)) = 1\,.$$

Due to this definition, the radius $r_i(t)$ ranges between $1/n$ and 1. To obtain a constant factor approximation for $r_i(t)$, we define $r_i^*(t)$ to be the value 2^{-k^*}, such that $k^* = k_0 - \lceil \log(4\sqrt{d}) \rceil$ and k_0 is the maximum integer in $\{0, 1, \ldots, \lfloor \log(n) \rfloor\}$, for which $weight(\mathcal{C}(p_i(t), 2^{-k_0})) \geq 2^{k_0}$ holds. The choice of k^* is explained in Section 4. Note that a cube $\mathcal{C}(p_i(t), r)$ is a ball with radius r with respect to the L_1-metric.

Walls around a point. For each point $p_i(t) \in \mathcal{P}(t)$, we consider a set of $\lfloor \log(n) \rfloor + 1$ nested cubes. In particular, for each $k \in \{-\lceil \log(4\sqrt{d}) \rceil, 1 - \lceil \log(4\sqrt{d}) \rceil, \ldots, \lfloor \log(n) \rfloor - \lceil \log(4\sqrt{d}) \rceil\}$, there is the cube $\mathcal{C}(p_i(t), 2^{-k})$. The side faces of such a cube form a wall around $p_i(t)$, which we call $W_{i,k}(t)$.

Range trees. We maintain two $(d+1)$-dimensional dynamic range trees denoted by T_1 and T_2. At any time, range tree T_1 is used to manage the current set of facilities, and T_2 stores the current set of clients. Both range trees are constructed in the same way. In the first d levels, the points are handled according to their coordinates and in the $(d+1)$-st level according to their special radii. Additionally, with each node v we store the number of all points contained in the subtree of v.

The dynamic data structure described in [3] supports all required properties of T_1 and T_2 efficiently. The movement of the points in our case is reflected by insertion and deletion operations on T_1 and T_2 upon an event. That means that the actual position of a point p_i is represented by its coordinates at the latest event it was involved in. Although its exact coordinates might slightly deviate between two events that involve p_i, we will show that, at any point of time t, the point $p_i(t) \in \mathcal{P}(t)$ is stored in the correct range with respect to the walls of all other points.

3 The Kinetic Data Structure

In this section, we describe the event queue and how an update of the KDS is processed.

Event Queue. To maintain an appropriate set of facilities, we have to update our KDS at certain points of time. More precisely, we perform an update each time a point $p_j(t)$ crosses a wall $W_{i,k}(t)$ with $-\lceil \log(4\sqrt{d}) \rceil \leq k \leq \lfloor \log(n) \rfloor - \lceil \log(4\sqrt{d}) \rceil$, of another point $p_i(t)$. In order to keep track of these events, we need another data structure beside the two range trees: For each dimension ℓ, $1 \leq \ell \leq d$, we store all n points and all $\mathcal{O}(n \log(n))$ wall faces which are orthogonal to the ℓ-th coordinate axis in a list sorted by the ℓ-coordinate. For each consecutive pair in each list, we keep up one certificate to certify the sorted order of the lists. The failure time of the certificate for any pair of consecutive objects is the first future time when these objects swap their places in their sorted list. We maintain the failure times of all certificates in one event queue. Certainly, it is not the case that each event implicates that a point crosses a wall of another point, but definitely every crossing of a wall is discovered by a failure of at least one certificate.

Handling an event. To get an initial solution for the kinetic FLP, we apply the algorithm of Mettu-Plaxton [9] on the input points. Afterwards, whenever an event occurs, we update both points involved according to their actual position in the two range trees. Then we update the event queue. All further steps are performed to keep up one invariant, which is fulfilled if the following conditions hold:

a) for each closed point $p_i(t) \in \mathcal{G}(t)$ there is an open point $p_j(t) \in \mathcal{F}(t)$ with $r_j^*(t) \leq r_i^*(t)$ in $\mathcal{C}(p_i(t), 4 \cdot r_i^*(t))$ and

b) for each open point $p_i(t) \in \mathcal{F}(t)$ there is no other open point $p_j(t) \in \mathcal{F}(t)$ with $r_j^*(t) \leq r_i^*(t)$ in $\mathcal{C}(p_i(t), 2 \cdot r_i^*(t))$.

We will show that, at each point of time t, if the invariant is fulfilled, $cost(\mathcal{F}(t))$ is at most a constant factor larger than the current optimal cost. Now, let us assume that the invariant is fulfilled by any point of time t when an event e occurs. Then the only possibility that the invariant gets violated is that e indicates that any point in $\mathcal{P}(t)$ crosses a wall of any other point $p_e(t)$. Thus, if e is not a wall crossing, handling e is finished after updating the event queue. Otherwise, we perform the following steps. At first, we update the radius $r_e^*(t)$. Note that $r_e^*(t)$ can be found by performing a binary search on its $\mathcal{O}(\log(n))$ possible values, in which each step of the binary search requires only one range query on both T_1 and T_2. Afterwards, we test if $p_e(t)$ violates the invariant by using a range query on T_1. If this is the case, we change the status of $p_e(t)$. As an effect of changing the radius or the status of one point, the invariant might be violated by many other points (e.g. their open facility has been closed).

Suppose that, due to an event e, the radius or the status of point $p_e(t)$ changed and its new radius is $r_e^*(t) = 2^{-k^*}$. First, we restore the invariant at all points with radius $2^{-(k^*+1)}$, to ensure that no point with radius less than or equal to $2^{-(k^*+1)}$ violates the invariant. Then, we handle the points with radius 2^{-k^*}, then the ones with radius $2^{-(k^*-1)}$, ..., up to radius $2^{\lceil \log(4\sqrt{d}) \rceil}$. Now, we describe how to proceed in general for any radius 2^{-k} (cf. Algorithm 3.1).

We define two cubical shells $S_1 := \mathcal{C}(p_e(t), 4 \cdot 2^{-(k-1)})$ and $S_2 := \mathcal{C}(p_e(t), 6 \cdot 2^{-(k-1)}) \setminus \mathcal{C}(p_e(t), 4 \cdot 2^{-(k-1)})$. Both cubical shells are divided into equally sized cubelets with radius 2^{-k}. Figure 1 (a) illustrates this decomposition in the plane for k and the next iteration $k-1$.

To guarantee that no open point with radius 2^{-k} violates the invariant, we perform the following test for each cubelet in S_1: Let m be the center point of the considered cubelet. If there is a facility with radius less than 2^{-k} in $\mathcal{C}(m, 3 \cdot 2^{-k})$, then close all facilities with radius 2^{-k} in $\mathcal{C}(m, 2^{-k})$. Note that there is at most one of them. The considered area around a cubelet is illustrated in Figure 1 (b).

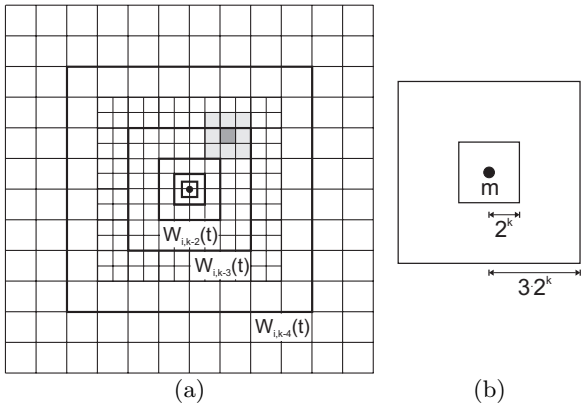


Figure 1: (a) Decomposition. (b) Tested area.

Then, we perform a similar test for each cubelet in both cubical shells (cf. Algorithm 3.1), to guarantee that the certificate of every closed point with radius 2^{-k} hold.

Algorithm 3.1 RESTORE$(p_e(t), k^*)$

1: **for** $k \leftarrow k^* + 1$ **downto** $-\lceil \log(4\sqrt{d}) \rceil$ **do**
2: define cubical shells $S_1 := \mathcal{C}(p_e(t), 4 \cdot 2^{-(k-1)})$, $S_2 := \mathcal{C}(p_e(t), 6 \cdot 2^{-(k-1)}) \setminus \mathcal{C}(p_e(t), 4 \cdot 2^{-(k-1)})$
3: **for** each cubelet C with radius 2^{-k} and center m_C in cubical shell S_1 **do**
4: **if** $\exists$ open facility with radius $< 2^{-k}$ in $\mathcal{C}(m_C, 3 \cdot 2^{-k})$ **then**
5: close all facilities with radius 2^{-k} in C
6: **for** each cubelet C with radius 2^{-k} and center m_C in cubical shell $S_1 \cup S_2$ **do**
7: **if** $\nexists$ open facility with radius $\leq 2^{-k}$ in $\mathcal{C}(m_C, 3 \cdot 2^{-k})$ **then**
8: open one point with radius 2^{-k} in C (if existing)

4 Quality and complexity

The difficulty in proving the correctness of maintaining the invariant is that both range trees contain outdated information. For any point of time t and any $p_i(t) \in \mathcal{P}(t)$, let $p_i^T(t)$ be the position of p_i stored in the range trees at time t. The following proposition shows that, at any time, every point is stored in the correct range with respect to the walls of all other points.

Proposition 1 *Let $p_i, p_j \in \mathcal{P}$, let $-\lceil \log(4\sqrt{d}) \rceil \leq k \leq \lfloor \log(n) \rfloor - \lceil \log(4\sqrt{d}) \rceil$ be an integer and let t be any point of time between two successive events which involve p_i and p_j. If and only if we have $p_j^T(t) \in \mathcal{C}(p_i^T(t), 2^{-k})$, then $p_j(t) \in \mathcal{C}(p_i(t), 2^{-k})$ is true as well.*

Proof. Let $t_1 < t$ be the latest point of time when p_i and p_j have been involved in one event. Furthermore, $p_j^T(t) \in \mathcal{C}(p_i^T(t), 2^{-k})$ implies that we have updated p_i and p_j at time t_1, such that $p_j^T(t_1) \in \mathcal{C}(p_i^T(t_1), 2^{-k})$. Because we have updated both p_i and p_j in the range trees at time t_1, $p_j(t_1) \in \mathcal{C}(p_i(t_1), 2^{-k})$ is also true. Now let us assume that we have $p_j^T(t) \in \mathcal{C}(p_i^T(t), 2^{-k})$ but $p_j(t) \notin \mathcal{C}(p_i(t), 2^{-k})$. Thus, there must be a point of time t_2 with $t_1 < t_2 < t$ when the point $p_j(t_2)$ crosses the wall $W_{i,k}(t_2)$. Then t_1 could not be the latest point of time when p_i and p_j have been involved in one event, leading to a contradiction. Analogously, we can show that $p_j^T(t) \notin \mathcal{C}(p_i^T(t), 2^{-k})$ implies $p_j(t) \notin \mathcal{C}(p_i(t), 2^{-k})$. □

It is easy to prove that the invariant is fulfilled as long as the algorithm does not call the subroutine RESTORE. Next, we show that the invariant is restored after each call of the subroutine RESTORE.

Proposition 2 *If no point with radius less than or equal to $2^{-(k^*+2)}$ violates the invariant before calling algorithm* RESTORE *with input parameter k^*, then this holds after running algorithm* RESTORE *as well.*

Proposition 3 *Let $p_e(t)$ be a point whose radius or status is changed due to an event e. Let $r_e^*(t) = 2^{-k^*}$ be the updated radius of $p_e(t)$. If the invariant is fulfilled before e and no point with radius less than or equal to $2^{-(\ell+1)}$ violates the invariant before running the outer* **for***-loop of algorithm* RESTORE *for $k = \ell$, where $-\lceil \log(4\sqrt{d}) \rceil \leq \ell \leq k^* + 1$, then, after running this* **for***-loop, no point with radius 2^ℓ violates the invariant.*

Proof. (sketch) Since p_e is updated in the range trees at time t, we have $p_e^T(t) = p_e(t)$. Due to Proposition 1, we know that $p_i(t) \in S_1$ if and only if $p_i^T(t) \in S_1$. Furthermore, due to Proposition 1, if we have $p_i(t) \in S_2$, then we also know that $p_i^T(t) \notin S_1$. Consequently, we have to consider the following five possible combinations for $p_i(t)$ and $p_i^T(t)$: i) $p_i(t) \in S_1$ and also $p_i^T(t) \in S_1$, ii) $p_i(t) \in S_2$ and also $p_i^T(t) \in S_2$, iii) $p_i(t) \in S_2$, but $p_i^T(t) \notin S_1 \cup S_2$, iv) $p_i(t) \notin S_1 \cup S_2$ and also $p_i^T(t) \notin S_1 \cup S_2$, and v) $p_i(t) \notin S_1 \cup S_2$, but $p_i^T(t) \in S_1 \cup S_2$.

Due to the tests performed for the cubelets in S_1 and S_2 and by making use of Proposition 1, we can prove each of the five cases by contradiction. □

Lemma 4 *The invariant is fulfilled after the algorithm has handled an event.*

The special radii. The authors in [1] showed how to approximate the radius $r_i(t)$ (cf. Section 2) by counting the number of points in a certain distance of $p_i(t)$. Our algorithm uses their approach, but we approximate the number of points in a distance 2^{-k}, for an integer k, by a cube with radius 2^{-k}.

Lemma 5 *Let k_0 be the maximum integer k, with $0 \leq k \leq \log(n)$, such that weight$(\mathcal{C}(p_i(t), 2^{-k})) \geq 2^k$. Then $\frac{1}{4\sqrt{d}} \cdot r_i(t) \leq 2^{-k_0} \leq 2 \cdot r_i(t)$ holds.*

In order to get $r_i(t) \leq r_i^*(t)$, we set $r_i^*(t) = 2^{-k^*} = 2^{-(k_0 - \lceil \log(4\sqrt{d}) \rceil)}$. This implies $r_i(t) \leq r_i^*(t) \leq 2^{3+\lceil \log(\sqrt{d}) \rceil} \cdot r_i(t)$.

Lemma 6 *There is a KDS maintaining an $\mathcal{O}(1)$-approximation for the uniform kinetic FLP in an arbitrary but fixed dimension d.*

Proof. Since for each point $p_i(t) \in \mathcal{P}(t)$ there is a facility $p_j(t)$ in $\mathcal{C}(p_i(t), 4 \cdot r_i^*(t))$, we get $D(p_i(t), p_j(t)) \leq \sqrt{d} \cdot 4 \cdot r_i^*(t) \leq \sqrt{d} \cdot 4 \cdot 2^{3+\lceil \log(\sqrt{d}) \rceil} \cdot r_i(t)$. Now, the lemma follows from the analysis in [9]. □

Generalization to the non-uniform FLP yields:

Theorem 7 *There is a KDS maintaining an $\mathcal{O}(1)$-approximation for the kinetic FLP for a set of n points in $\mathbb{R}^d$, where d is arbitrary but fixed. The space requirement is $\mathcal{O}(n(\log^d(n) + \log(nR)))$, where R is the ratio of the product of the maximum maintenance cost and demand to the product of their corresponding minimum values. In the case that each trajectory can be described by a bounded degree polynomial, the total number of updates is $\mathcal{O}(n^2 \log^2(nR))$, each requiring $\mathcal{O}(\log^{d+1}(n) \cdot \log(nR))$ time. A flight plan update involves $\mathcal{O}(\log(nR))$ certificates and requires $\mathcal{O}(\log^2(nR))$ time.*

References

[1] M. Bădoiu, A. Czumaj, P. Indyk, and C. Sohler. Facility location in sublinear time. *Proc. 32nd Internat. Colloq. Automata Lang. Program.*, 3580:866–877, 2005.

[2] J. Basch and L. Guibas and J. Hershberger. Data structures for mobile data. *J. Algorithms*, 31(1): 1–28, 1999.

[3] J. Basch and L. Guibas and L. Zhang. Proximity problems on moving points. *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pp. 344–351, 1997.

[4] S. Bespamyatnikh, B. Bhattacharya, D. Kirkpatrick, and M. Segal. Mobile facility location. In *Proc. 4th Internat. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pp. 46–53, 2000.

[5] B. Degener, J. Gehweiler, and C. Lammersen. The Kinetic Facility Location Problem. *Technical Report tr-ri-08-288*, 2008.

[6] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Discrete mobile centers. *Discrete Comput. Geom.*, 30(1):45–63, 2003.

[7] L. Guibas. Modeling Motion. *Handbook of Discrete and Computational Geometry*, pp. 1117–1134, 2004.

[8] J. Hershberger. Smooth kinetic maintenance of clusters. *Proc. Annu. ACM Sympos. Comput. Geom.*, pp. 48–57, 2003.

[9] R. R. Mettu and C. G. Plaxton. The online median problem. In *SIAM J. Comput.*, 32(3): 816-832, 2003.

Probabilistic Matching of Polygons*

Helmut Alt† Ludmila Scharf† Daria Schymura†

Abstract

We analyze a probabilistic algorithm for matching plane compact sets with sufficiently nice boundaries under translations and rigid motions (rotation and translation). Given shapes A and B, the algorithm computes a transformation t such that the area of overlap of $t(A)$ and B is close to maximal with high probability. We give a time bound that does not depend significantly on the number of vertices in the case of polygons.

1 Introduction

Shape matching is a problem that occurs in various areas of computer science and in various flavors, particular in computer vision.

We model shapes as bounded polygonal regions in the plane. Given two shapes A and B, as well as a set of transformations F and a distance measure d, we look for the transformation $t \in F$ such that $t(A)$ and B match optimally with respect to d. Two shapes are similar if there is a transformation t for that the distance between $t(A)$ and B is small.

This is a well-studied problem in the case of rigid motions and the Hausdorff distance, if the shapes are modeled as sets of line segments, for example. For a survey on the topic, see [3].

We study the case where F is the set of translations or rigid motions (rotation and translation) in the plane and the distance measure is the area of the symmetric difference, that is the area that belongs to exactly one of the shapes. Minimizing the area of the symmetric difference under translations or rigid motions is the same as maximizing the area of overlap, and that is what we will speak about from now on. The area of overlap is a well-known similarity measure, which is insensitive to noise.

There are efficient algorithms for polygons that maximize the area of overlap under translations. Mount et al. [8] show that the maximal area of overlap of a simple n-polygon with a translated simple m-polygon can be computed in $O(n^2m^2)$ time. Recently, Cheong et al. [7] gave a general probabilistic framework that computes an approximation with prespecified absolute error ε in $O(m + (n^2/\varepsilon^4)\log(n)^2)$ time for translations and $O(m+(n^3/\varepsilon^4)\log(n)^5)$ time for rigid motions. De Berg et al. [6] consider the case of convex polygons and give a $O((n+m)\log(n+m))$ time algorithm that maximizes the area of overlap. Alt et al. [2] give a linear time constant factor approximation algorithm for minimizing the area of the symmetric difference under translations and homotheties (scaling and translation).

Surprisingly little is known about maximizing the area of overlap under rigid motions and similarities.

Here, we analyze a probabilistic algorithm that approximates the maximal area of overlap under translations and rigid motions. Given an allowable error ε and a desired probability of success p, we show bounds on the required number of random experiments, guaranteeing that the absolute difference between approximation and optimum is at most ε with probability at least p.

This algorithm is a special case of a probabilistic algorithmic scheme for approximating an optimal match of compact sets under a subgroup of affine transformations. Alt and Scharf [4] analyzed an instance of this algorithmic scheme that compares polygonal curves under translations, rigid motions, and similarities.

2 The algorithm

The idea of the algorithm is to draw random points from the shapes, to compute a transformation that maps the points onto each other, and to keep this transformation, called a "vote", in mind. This is repeated very often; in each step, we grow our collection of "votes" by one. Clusters of "votes" indicate transformations that map large parts of the shapes onto each other. The parameter δ adjusts the clustering size. Now we state the algorithm for translations.

Given: shapes A and B, integer n, positive real δ.

1. Do the following experiment n times:
   Draw uniformly distributed random points $a \in A$ and $b \in B$. Give one "vote" to the unique translation that maps a onto b.
2. Determine and return one of the translations whose δ-neighborhood obtained the most "votes".

The term δ-neighborhood refers to the maximum norm; we identify each translation with its translation

*This research was supported by the European Union under contract No. FP6-511572, Project PROFI.

†Institute of Computer Science, Freie Universität Berlin, {alt,scharf,schymura}@inf.fu-berlin.de

vector and equip the translation space $\mathbb{R}^2$ with the maximum norm.

The algorithm captures the intuitive notion of matching. Translations that many pairs of points vote for should be "good" translations since many points from A are mapped onto points from B. Figure 1 illustrates this idea.

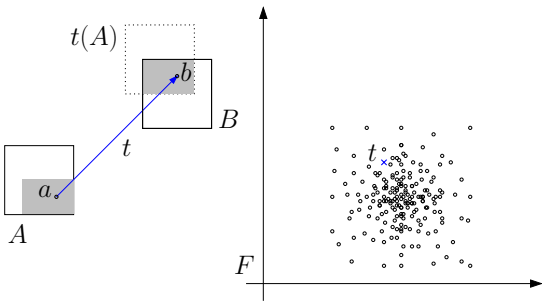


Figure 1: We compare two copies of a square under translations. The area of overlap of $t(A)$ and B corresponds to the chance of choosing a point pair $(x, y) \in A \times B$ such that $y - x = t$.

Now we explain the algorithm for rigid motions. The space of rigid motions R is given as $[-\pi, \pi) \times \mathbb{R}^2$, equipped with the maximum norm. A point $(\alpha, t) \in R$ denotes the rigid motion

$$x \mapsto M_\alpha x + t, \quad M_\alpha = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}.$$

For matching under rigid motions, we draw in each step uniformly distributed an angle α and random points $a \in A$ and $b \in B$. We give one "vote" to the unique rigid motion with counterclockwise rotation angle α that maps a onto b, namely

$$x \mapsto M_\alpha x + (b - M_\alpha a).$$

When speaking of transformations, we refer to translations and rigid motions.

The algorithm does not require the shapes to be polygons; it works for measurable sets in $\mathbb{R}^2$, provided there is a method to draw uniformly distributed random points from them and the density function is Lipschitz continuous (see Section 4.3). The latter is fulfilled, for example, if the shapes' boundaries are unions of piecewise differentiable simple closed curves. The algorithm can be directly applied on bitmap data as well.

3 Main result

We study the transformation space with the probability distribution that is implicitly given by the random experiment and the fact that we draw the random points from the shapes in a uniformly distributed way.

The distribution of "votes" in the transformation space approximates the probability distribution that results from the experiment; the fraction of "votes" that fall in a set approximates its probability. The output of the algorithm is a transformation t whose δ-neighborhood has approximately highest probability.

The probability Pr of an event E and the density function g are by definition related by $Pr(E) = \int_E g(s)ds$. If the density function is uniformly continuous and δ is small enough, the transformations whose δ-neighborhood have highest probability and the transformations at which the density function is maximal are the same. Then, the output is a transformation t at which the density function is close to the maximum with high probability.

It turns out that, in our case, the density function is the function mapping a transformation vector to the area of overlap of the transformed shape A and B divided by a constant.

Let μ be the Lebesgue measure on $\mathbb{R}^2$, which for polygons equals the area. We always assume the shapes not to be degenerate.

Lemma 1 *The density function of the probability distribution on the translation space that results from the experiment is given by*

$$f(t) = \frac{\mu(t(A) \cap B)}{\mu(A)\ \mu(B)}.$$

In the case of rigid motions, the density function on R is given by

$$g(r) = \frac{\mu(r(A) \cap B)}{2\pi\ \mu(A)\ \mu(B)}.$$

The main result is the following approximation theorem, which states that, for each allowable error ε and each desired probability of success p, there is a number of experiments N guaranteeing approximation of the maximal area of overlap with error at most ε and with probability at least p.

Theorem 2 *Let t^* be a transformation that is output of the algorithm (described in Section 2) and t^{opt} a transformation that maximizes the area of overlap of A and B, when applied to A. Let $\varepsilon > 0$ and $p < 1$. There are a positive real $\delta = O(\varepsilon)$ and an integer N such that with probability at least p*

$$|\mu(t^*(A) \cap B) - \mu(t^{\text{opt}}(A) \cap B)| < \varepsilon$$

if N is the number of performed random experiments.

In the case of translations,

$$N = O(c/\varepsilon^6 \times \log(c(p-1)/\varepsilon^6)).$$

In the case of rigid motions,

$$N = O(C/\varepsilon^8 \times \log(C(p-1)/\varepsilon^8)).$$

where $c = \mu(A)^2\mu(B)^2\Delta^4$, $C = \mu(A)^2\mu(B)^2\Delta^6 D^6$, Δ *is the maximum of the boundary lengths of* A *and* B, *and* D *is the maximum of the diameters of* A *and* B.

A sketch of the proof is given in Section 4.4.

In the case of polygons, after triangulation, we can draw random points in logarithmic time. The arrangement of δ-spheres whose center is a "vote" can be built and traversed within $O(N^2)$ and $O(N^3)$, respectively. The runtime of the algorithm is bounded by $O(N^2 + N\log n + n\log n)$ for translations and by $O(N^3 + N\log n + n\log n)$ for rigid motions, if N is the required number of experiments and n is total number of vertices of the polygons. In contrast to all other known approaches, the runtime does not depend significantly on the number of vertices.

4 Analysis of the algorithm

For a transformation t, let $B_\delta(t)$ be the δ-neighborhood of t in the maximum norm. Recall that $B_\delta(t)$ is two-dimensional in the case of translations and three-dimensional in the case of rigid motions. Denote by $X_n^\delta(t)$ the fraction of "votes" that lies δ-close to t after n experiments. Intuitively, it is not surprising that the following convergence relations hold

$$X_n^\delta(t) \quad \stackrel{n\to\infty}{\longrightarrow} \quad Pr(B_\delta(t)) \quad \stackrel{\delta\to 0}{\longrightarrow} \quad \mu(B_\delta(t))\; g(t),$$

where the first convergence is in probability.

4.1 Estimating the probability of a fixed δ-ball

The estimate $X_n^\delta(t)$ is called naive estimator in the theory of density estimation [9]. The next lemma states that for each transformation t the estimate $X_n^\delta(t)$ is close to $Pr(B_\delta(t))$ with high probability; it can be proven by using the Chernoff bound, as stated in [7].

Lemma 3 *For each transformation* t *and for all* $0 < \varepsilon < 1$ *the following holds*

$$Pr(|X_n^\delta(t) - Pr(B_\delta(t))| > \varepsilon) < 2e^{-\frac{\varepsilon^2 n}{2}}.$$

Later, we will have to show that the output of the algorithm is a transformation that approximately maximizes $Pr(B_\delta(t))$. The latter is not an obvious corollary of Lemma 3 since the output transformation t is a random vector depending on the sequence of experiments.

4.2 Density function

In this section we show the proof of Lemma 1 for rigid motions; the proof for translations is a simpler version of it.

We will use the following special case of a transformation formula for density functions of random variables, which can be found in introductory books about probability theory.

Theorem 4 *Let* $X : \mathbb{R}^n \to \mathbb{R}^n$ *be a random variable with density function* f_X *and* $h : \mathbb{R}^n \to \mathbb{R}^n, h : x \mapsto Mx$ *a linear map with* $\det(M) \neq 0$. *Then* $h\circ X$ *has the density function* $f_{h\circ X}(y) = f_X(M^{-1}y)\;|\det(M^{-1})|$.

Proof. (of Lemma 1) Our random experiment consists in drawing uniformly distributed points from $\Omega = I \times A \times B$ where $I = [-\pi, \pi)$. We are interested in the density function f_X of the random variable

$$X : \Omega \to R, \quad X : (\alpha, a, b) \mapsto (\alpha, b - M_\alpha a).$$

Drawing the counterclockwise rotation angle uniformly distributed in I corresponds to the random variable id_I with density function $f_I(\alpha) = \frac{1}{2\pi}$.

Determining the translation vector t depends on the rotation angle α. First, we compute the density function f_α of the random variable X_α that is defined as follows:

$$X_\alpha : A \times B \to \mathbb{R}^2, \quad (a, b) \mapsto b - M_\alpha a.$$

Understanding f_α as conditional density $f_X(\alpha, \cdot)$ on $\mathbb{R}^2$ gives then

$$f_X(\alpha, t) = f_I(\alpha)\; f_\alpha(t).$$

Therefore it suffices to compute f_α.

The function $h : \mathbb{R}^4 \to \mathbb{R}^4, h : (a, b) \mapsto (a, b - M_\alpha a)$ is a linear map with determinant 1. Let $\pi : \mathbb{R}^4 \to \mathbb{R}^2$ be the projection on the third and forth coordinate, then $X_\alpha = \pi \circ h \circ id_{A\times B}$. For a set $E \subset X$, let $\chi_E : X \to \{0, 1\}$ be its characteristic function that is one iff $x \in E$. We know $f_{id_{A\times B}}(a, b) = \chi_A(a)\chi_B(b)/(\mu(A)\mu(B))$. Using Theorem 4, we get

$$f_{h\circ id_{A\times B}}(a, b) = \chi_{A\cap(M_{-\alpha}(B-b))}(a)/(\mu(A)\mu(B)).$$

Now we can compute f_α, which proves the theorem:

$$\begin{aligned} f_\alpha(t) &= \int_A f_{h\circ id_{A\times B}}(a, t)da \\ &= \mu((M_\alpha A + t) \cap B)/(\mu(A)\mu(B)). \end{aligned}$$

□

4.3 Lipschitz continuity of the density function

A function f from a metric space M to $\mathbb{R}$ is called Lipschitz continuous if there is a constant L such that for all $x, y \in M$ holds that $\|x - y\| < \delta$ implies $|f(x) - f(y)| < L\delta$.

Denote by μ_δ the Lebesgue measure of the δ-neighborhood of a transformation t in the metric induced by the maximum norm. The number μ_δ does

not depend on t since the Lebesgue measure is invariant under translations and rotations. For translations, $\mu_\delta = 4\delta^2$ and for rigid motions, $\mu_\delta = 8\delta^3$. We are interested in the density functions to be Lipschitz continuous because then holds

$$|Pr(B_\delta(t)) - \mu_\delta\ f(t)| \leq L\mu_\delta\delta.$$

Translations. Let $\Delta_{A,B}$ be the sum of the boundary lengths of A and B.

It is easy to show, that for the density function f on the translation space, $\sqrt{2}\Delta_{A,B}/(\mu(A)\mu(B))$ is a Lipschitz constant. Observe that the constant depends heavily on the shapes.

Rigid motions. Let D_B be the minimum radius of a ball around the origin that contains B and D_A the analogue for A. Standard geometric arguments show that for the density function g on the space of rigid motions, $(2D_B + D_A)\Delta_{A,B}/(2\pi\mu(A)\mu(B))$ is a Lipschitz constant.

4.4 Proof sketch of the main result

We have already seen that for fixed t

$$X_n^\delta(t) \underset{\text{in prob.}}{\overset{n\to\infty}{\longrightarrow}} Pr(B_\delta(t)) \overset{\delta\to 0}{\longrightarrow} \mu_\delta\ g(t).$$

Obviously, two errors are involved in the approximation process. One we call the sampling error; it becomes smaller if the number of experiments increases and can be bounded by the Chernoff inequality. The other we call the smoothing error; it becomes smaller if δ decreases and can be bounded by the Lipschitz continuity of the density function. Instead of smoothing, we could discretize the shapes and would end up with a discretization error added to the sampling error. This would simplify the analysis a little but could not be generalized so nicely to other transformation groups.

Now we need to analyze what happens if the transformation vector is determined by the sequence of random experiments, namely the vector whose δ-neighborhood obtains the most "votes", and thus is a random vector itself.

The output of the algorithm can be modeled as random variable

$$Z_n^\delta = \max_{t\in\mathbb{R}^2} X_n^\delta(t).$$

Let $S = (s_1, \ldots, s_n)$ be a sequence of transformations from the random experiments. Consider the arrangement induced by the boundaries of $B_\delta(s_1), \ldots, B_\delta(s_n)$, which are the δ-spheres of the points in S. The depth of a cell is defined as the number of $B_\delta(s_i)$ it is contained in. The candidates for the output of the algorithm are the transformations corresponding to the deepest cells in this arrangement. A transformation t lies in the intersection of k of the neighborhoods if and only if its neighborhood contains k "votes".

The next lemma can be proven using an idea of [7].

Lemma 5 *Let V be a set of points such that V contains for each cell of the arrangement induced by the δ-spheres with centers in S one point of its lowest-dimensional face. There is such a V that contains at most n^2 points in the case of translations and n^3 in the case of rigid motions. For each $\varepsilon > 0$ and all $n \geq \frac{6}{\varepsilon} + 2$, it holds that*

$$Pr(\exists t \in V : |X_n^\delta(t) - Pr(B_\delta(t))| > \varepsilon)$$

is less than $2n^2e^{-\varepsilon^2(n-2)/8}$ in the case of translations and less than $2n^3e^{-\varepsilon^2(n-3)/16}$ in the case of rigid motions.

Using Lemma 5 and the Lipschitz continuity of the density functions, the main result can be proven.

We note that the proof provides explicit bounds for the required number of experiments to ensure approximation with error at most ε with probability at most p.

References

[1] H.-K. Ahn, O. Cheong, C.-D. Park, C.-S. Shin and A. Vigneron. *Maximizing the overlap of two planar convex sets under rigid motions.* Computational Geometry, 37(1):3–15, 2007.

[2] H. Alt, U. Fuchs, G. Rote and G. Weber. *Matching convex shapes with respect to the symmetric difference.* Algorithmica, 21:89–103, 1998.

[3] H. Alt and L. Guibas. *Discrete Geometric Shapes: Matching, Interpolation, and Approximation. A Survey.* Handbook of Computational Geometry, eds. J.-R. Sack and J. Urrutia, pages 121–153. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 1999.

[4] H. Alt and L. Scharf. *Shape matching by random sampling.* Technical Report B 08-02, Freie Universität, 2007.

[5] H. Alt, L. Scharf and S. Scholz *Probabilistic matching of sets of Polygonal curves.* Proceedings of the 22nd European Workshop on Computational Geometry (EWCG), 107–110, March 2006, Delphi, Greece.

[6] M. de Berg, O. Devillers, M.J. van Kreveld, O. Schwarzkopf and M. Teillaud. *Computing the maximum overlap of two convex polygons under translations.* Theory of computing systems, 31:613–628,1998.

[7] O. Cheong, A. Efrat and S. Har-Peled. *Finding a guard that sees most and a shop that sells most.* Discrete and Computational Geometry, 37(4):545–563, 2007.

[8] D.M. Mount, R. Silverman and A.Y. Wu. *On the area of overlap of translated polygons.* Computer Vision and Image Understanding: CVIU, 64(1):53–61, 1996.

[9] B.W. Silverman. *Density Estimation for Statistics and Data Analysis.* Chapman & Hall, 1986, New York.

Author Index