Convex Tours of Bounded Curvature

Jean-Daniel Boissonnat, Jurek Czyzowicz, Olivier Devillers, Jean-Marc Robert, Mariette Yvinec

▶ **To cite this version:**

Jean-Daniel Boissonnat, Jurek Czyzowicz, Olivier Devillers, Jean-Marc Robert, Mariette Yvinec. Convex Tours of Bounded Curvature. [Research Report] RR-2375, INRIA. 1994. inria-00074301

HAL Id: inria-00074301
https://inria.hal.science/inria-00074301v1

Submitted on 24 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Convex Tours of Bounded Curvature

Jean-Daniel Boissonnat, Jurek Czyzowicz, Olivier Devillers,
Jean-Marc Robert and Mariette Yvinec

N° 2375

Octobre 1994

PROGRAMME 4

Robotique,

image

et vision

1994

ISSN 0249-6399

Convex Tours of Bounded Curvature

Jean-Daniel Boissonnat*, Jurek Czyzowicz**, Olivier Devillers***,
Jean-Marc Robert**** and Mariette Yvinec*****

Programme 4 — Robotique, image et vision
Projet Prisme

Rapport de recherche n° 2375 — Octobre 1994 — 16 pages

Abstract: We consider the motion planning problem for a point constrained to move along a smooth closed convex path of bounded curvature. The workspace of the moving point is bounded by a convex polygon with m vertices, containing an obstacle in a form of a simple polygon with n vertices. We present an $O(m+n)$ time algorithm finding the path, going around the obstacle, whose curvature is the smallest possible.

Key-words: Computational geometry, Motion planning, Non holonomy

(Résumé : tsvp)

This work has been supported in part by the ESPRIT Basic Research Actions Nr. 7141 (ALCOM II) and Nr. 6546 (PROMotion), NSERC, FCAR and FODAR.

*INRIA, E-mail: Jean-Daniel.Boissonnat@sophia.inria.fr.
**Dép. d'informatique, Université du Québec à Hull, Canada. E-mail: czyzowicz@uqah.uquebec.ca.
***INRIA, E-mail: Olivier.Devillers@sophia.inria.fr.
****Dép. d'informatique et de mathématique, Université du Québec à Chicoutimi, Canada. E-mail: jmrobert@uqac.uquebec.ca.
*****INRIA, and CNRS URA 1376, I3S, Sophia-Antipolis, France. E-mail: Mariette.Yvinec@sophia.inria.fr.

Unité de recherche INRIA Sophia-Antipolis
2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex (France)
Téléphone : (33) 93 65 77 77 – Télécopie : (33) 93 65 77 65

Circuits convexes de courbure bornée

Résumé : Nous recherchons pour un point une trajectoire lisse, convexe et de courbure bornée. Le point se déplace à l'intérieur d'un polygone convexe à m cotés contenant un obstacle constitué d'un polygone simple de taille n. Nous proposons un algorithme de complexité $O(m+n)$ déterminant un circuit autour de l'obstacle de courbure minimale.

Mots-clé : Géométrie algorithmique, Planification de trajectoires, Non holonomie.

1 Introduction

Consider the problem of moving a point robot in the interior of a convex polygon containing a single obstacle. We are looking for a smooth, closed, convex, curvature-constrained path of the point around the obstacle. No source or target position of the point are specified.

The problem of planning the motion of a robot subject to kinematic constraints has been studied in numerous papers in the last decade (cf. [Lat91], [SS90]). For example, Reif and Sharir [RS85] studied the problem of planning the motion of a robot with a velocity bound amidst moving obstacles in two and three-dimensional space. Ó'Dúnlaing [Ó'D87] presented an exact algorithm solving the one-dimensional kinodynamic motion planning problem whereas Canny, Donald, Reif and Xavier [CDRX88] gave the first approximation algorithm solving the two and three dimensional kinodynamic motion planning for a point amidst polyhedral obstacles.

Another aspect of the motion planning problem in the plane consists in finding paths under curvature constraints. Dubins [Dub57] characterized shortest curvature constrained paths in the Euclidean plane without any obstacle. More recently, Fortune and Wilfong [FW88] gave a decision procedure to verify if the source and target placement of a point robot may be joined by a curvature constrained path avoiding the polygonal obstacles. Their procedure has time and space complexity $2^{O(poly(n,m))}$, where n is the number of obstacle vertices, and m is the number of bits required to specify the positions of these vertices. Jacobs and Canny [JC89] gave an algorithm computing an approximate curvature constrained path, and Wilfong [Wil88] designed an exact algorithm for the case where the curvature constrained path is limited to some fixed straight "lanes" and circular arc turns between the lanes. Finally, Švestka [Š93] applied the random approach introduced by Overmars [Ove92] to compute curvature constrained paths for car-like robots.

Besides heuristic and approximating approaches, an exact algorithmic solution seems to be difficult to find for the general case. An interesting direction of research is to design exact algorithms for some variants of the problem. In this paper, we give an efficient solution for the problem of computing a smooth, convex path going around a single polygonal obstacle with n vertices inside a convex polygon with m vertices. We design an $O(n+m)$ time and space algorithm finding a path of smallest curvature. The idea of the algorithm is to compute the curvature constraints imposed by each obstacle vertex. The maximal such constraint is then used to compute the curve, which must surround the entire obstacle.

We mention some extensions of this solution for the case of numerous obstacles, and for the case of obstacles coming as queries in a dynamic setting.

2 Preliminaries

Let $E \subset \mathbb{R}^2$ be a convex polygon with m vertices and let $I \subset E$ be a simple polygon with n vertices. The region $E \setminus int(I)$ represents the *workspace* W in which the point robot can move. A function $p : [0, L] \to W$ is a *smooth path* if $p(r) = (x_p(r), y_p(r))$ and the functions $x_p, y_p : [0, L] \to \mathbb{R}$ are continuous with continuous derivative. A smooth path p is *closed* if $p(0) = p(L)$ and its right derivative at point 0 is equal to its left derivative at point L. As any smooth path has finite length, we can assume that p is parameterized by arc length. Let $\phi_p(r)$ be the direction of the tangent to $p(r)$. The *curvature* of p at a point r can be defined as $\lim_{r' \to r} \frac{|\phi_p(r) - \phi_p(r')|}{|r - r'|}$. A path p has its *average curvature* bounded by a constant κ if $|\phi_p(r_2) - \phi_p(r_1)| \leq \kappa |r_2 - r_1|$ for all r_1, r_2.

A curvature bounded closed path p is a *tour* of I in E if the bounded region of E, delimited by the Jordan curve p, is convex and contains I. Note that the points of boundaries of E and I are allowed to lie on the tour. Finally, a tour is *optimal* if the bound on its curvature is the smallest possible.

The main problem considered in this paper can be formulated as follows. Find an optimal tour of I in E. We first consider the degenerated case where the internal polygon I is a single point.

Lemma 1 *For a given convex polygon E and a point v inside E, let C denote a circle of radius r contained in E, passing through v, and tangent to the boundary of E in two points p_1 and p_2. If the arc $p = p_1 v p_2$ of C is not greater than a semicircle, then any tour of v in E contains a point with curvature at least equal to $1/r$ (see Fig. 1).*

Proof: Any tour of v in E must intersect arc p. Let t denote such a tour. When we translate p along the bisector of the angle defined by the two lines tangent to C at p_1 and p_2, the extremities of p remain outside or on the boundary of E. Let p' be the furthest translated position of p, at which t is tangent to p' at a point x (c.f. Fig. 1). The curvature of t at x is at least equal to $1/r$. □

Following the above lemma, suppose that we have a circle C inscribed in the convex polygon E, being tangent to E in two points p_1 and p_2. We say that C is the *critical circle* of v in E, if arc $p_1 v p_2$ is not greater than a semicircle. Arc $p_1 v p_2$ is called the *critical arc* of v in E. Observe that only points lying outside the largest circle inscribed in E admit critical arcs.

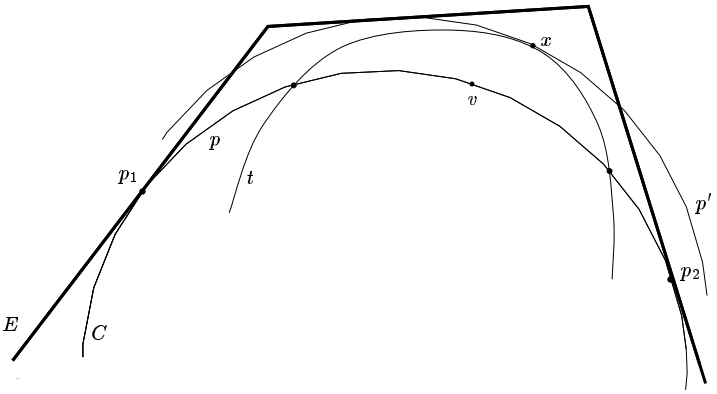

Figure 1: An optimal tour of a point.

3 Computing tours

3.1 The Case of Given Curvature

Consider the problem of computing, if one exists, a tour of I in E with curvature bounded by some given constant κ. We present in this section an algorithm solving this problem in $O(m+n)$ time. The algorithm proceed by computing a "maximal tour" in E with curvature bounded by κ.

Let S be the set of all circles of radius $1/\kappa$ inscribed in E, and tangent to E in at least two points. Note that S contains at most m circles. The curve ζ, formed by the boundary of the convex hull of S, is a smooth closed path with curvature bounded by κ. Such a path ζ is called a *maximal path* in E. It follows from the proof of Lemma 1 that the convex region bounded by ζ contains any smooth, closed path internal to E with curvature bounded by κ (see Fig. 2). Hence, if ζ is not a tour of I in E, there exists no tour of I in E with curvature bounded by κ.

Before we turn our attention to the algorithm verifying the existence of a tour of given curvature, we introduce some useful concepts. Consider the medial axis of E [Pre77]. Since E is a convex polygon, its medial axis corresponds to a tree. Each internal vertex x of this tree is the center of a circle tangent to three edges of E. This circle is called a *Voronoi circle*. We assign to x a weight $w(x)$ corresponding to the radius of the Voronoi circle. Thus, $w(x)$ represents the distance between x and the boundary of E. This weighted tree, rooted at its vertex with the largest weight,

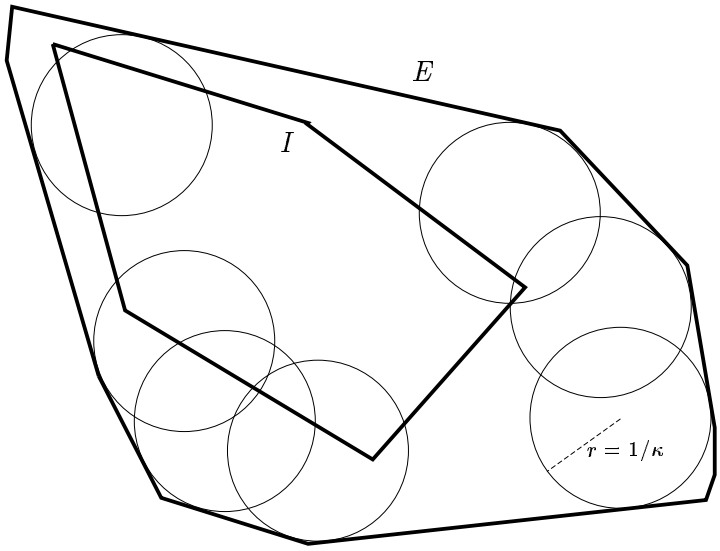

Figure 2: There is no tour of I in P with curvature bounded by κ

is called the *skeleton tree* and is denoted SkT(E). It follows from the definition of the medial axis that each edge of SkT(E) is a straight line segment belonging to the bisector of some two edges of E. It follows also from the definition that each vertex of SkT(E) has at least two descendants. Finally, we can easily prove that the weight of any vertex in SkT(E) is greater than the weights of its descendants.[1] This property will be crucial for our algorithms.

We are now ready to present how to compute the maximal path ζ.

Lemma 2 *Given the skeleton tree SkT(E), the maximal path ζ in E with curvature bounded by κ can be computed in $O(k)$ time, where k is the size complexity of the path.*

Proof: Perform a tree traversal on SkT(E). Each time a vertex x is visited, such that $w(parent(x)) \geq 1/\kappa > w(x)$, output a circle with radius $1/\kappa$, tangent to the boundary of E, and centered on the edge joining x and $parent(x)$. This circle can be computed easily once we know the edges of E defining the edge joining x to $parent(x)$. Then, the subtree of SkT(E) rooted at x is pruned and the traversal

[1] The root may have the same weight as one of its children if E has two parallel edges.

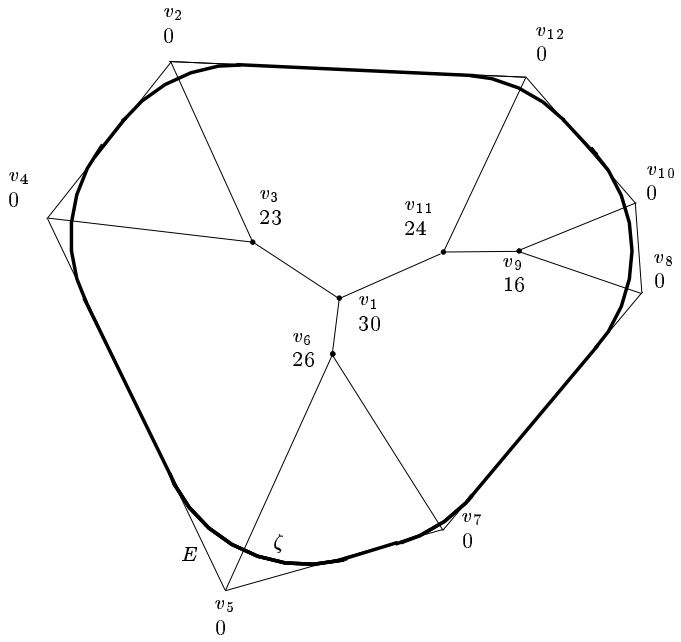

Figure 3: A skeleton tree and a maximal path ζ of bounded curvature. The arcs of ζ of radius 20 are centered on v_3v_2, v_3v_4, v_6v_5, v_6v_7, $v_{11}v_9$ and $v_{11}v_{12}$.

continues from $parent(x)$. In this way, all the k circles with radius $1/\kappa$ inscribed in E are found in order of their appearance on ζ. Hence, joining two consecutive circles by their supporting segment, we get the maximal path ζ, bounding the convex hull of the set circles. The $O(k)$ time complexity of the algorithm follows from the fact that the number of vertices visited during the transversal of $\mathrm{SkT}(E)$ is in $O(k)$. □

It should be obvious now how to determine if there exist a tour of I in E with curvature bounded by κ. First, compute the medial axis of E in $O(m)$ time [AGSS89]. Then, compute the maximal path ζ. Finally, determine if I lies completely inside ζ. This latter step can be done easily in $O(n+k)$ time where k is the complexity of ζ. Hence, our algorithm determines if there exists a tour of I in E with curvature bounded by κ in $O(m+n)$ time.

3.2 An Algorithm Computing Optimal Tours

Consider the problem of computing an optimal tour of I in E. An algorithm solving this problem can be sketched as follows. First, find the largest circle inscribed in E. If this circle contains I, it is obviously the optimal tour. Otherwise, for each vertex of I lying outside the largest inscribed circle, compute the radius of its critical circle in E. The minimum of these radii determines the curvature of the optimal tour. Once the curvature of the optimal tour is known, a tour can be computed as we described in the previous section. We present in this section how to implement this algorithm optimally in $O(m+n)$ time.

We first present the data structures used by the algorithm. Let $CH(I)$ be the list of the vertices of the convex hull of I, given in counterclockwise order. Now, let $Arcs(E)$ be the list of arcs defined as follows. Consider the Voronoi circles associated with the internal vertices of $\mathrm{SkT}(E)$. The tangent points of these circles with the boundary of E partition each circle into at least three arcs. Each of these arcs is put in $Arcs(E)$ if it is less than a semicircle. We also put in $Arcs(E)$ the leaves of $\mathrm{SkT}(E)$. These points represent degenerated arcs. The elements of $Arcs(E)$ are ordered such that the first endpoints of the arcs appear in counterclockwise order on the boundary of E (see Fig. 4). In the next lemma, we show how to build list $Arcs(E)$ efficiently.

Lemma 3 *$Arcs(E)$ can be generated in $O(m)$ time and space.*

Proof: Perform a tree traversal on $\mathrm{SkT}(E)$. The traversal can be oriented such that the children of any node are visited in counterclockwise order. An arc is produced each time a vertex x is visited from its parent v. This arc is less than a semicircle, centered at v, and tangent to the two edges of E whose bisector contains the edge vx of $\mathrm{SkT}(E)$. Furthermore, a degenerated arc is produced if x is a leaf of $\mathrm{SkT}(E)$.

To see that the arcs are produced in the right order, observe that the tree traversal can be performed by moving a point z continuously along the edges of $\mathrm{SkT}(E)$. Let $\pi(z)$ be the orthogonal projection of z on the edge of E belonging to the Voronoi cell on the right-hand side of z with respect to the direction of the traversal. Since $\mathrm{SkT}(E)$ corresponds to the medial axis of a convex polygon, $\pi(z)$ moves continuously around the boundary of E in the counterclockwise direction. Now, consider the arc computed while z traverses the edge vx of $\mathrm{SkT}(E)$. By construction, the first endpoint of this arc corresponds to $\pi(z)$ when z coincides with v. Thus, the arcs are produced during the traversal of $\mathrm{SkT}(E)$ such that the first endpoints of the arcs appear in counterclockwise order on the boundary of E.

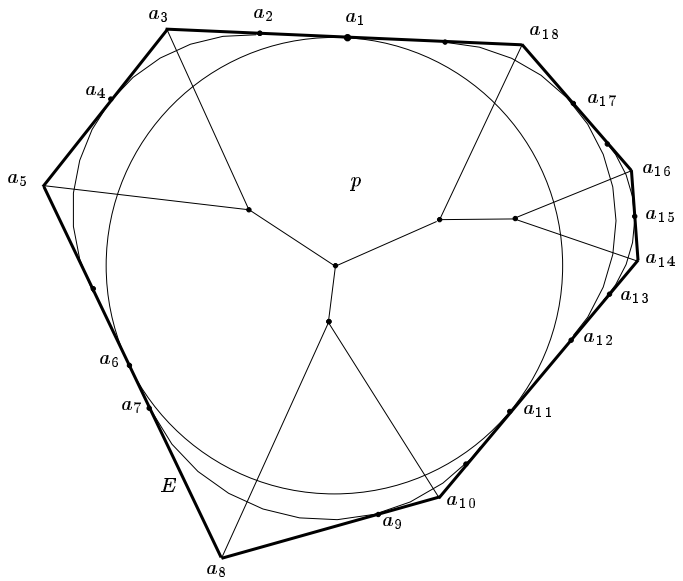

Figure 4: $Arcs(E)$ is determined according to order of arcs' first endpoints

The $O(m)$ time and space complexities of the algorithm follow from the fact that $\mathrm{SkT}(E)$ has at most $2m-2$ vertices. □

The elements of $CH(I)$ and $Arcs(E)$ will be traversed by the algorithm according to the circular counterclockwise order around a point p inside $CH(I)$. The choice of p do not affect the order within $CH(I)$ or within $Arcs(E)$, but it may affect the position of an element of $CH(I)$ with respect to an element of $Arcs(E)$. Variable V will denote the current element of $CH(I)$ and variable A will denote the current element of $Arcs(E)$. We say that vertex V is *before* arc A, if it precedes the first endpoint of A in the circular order around p. V is *after* A if it succeeds the second endpoint of A in this order. For V situated neither before nor after A, V is *inside* A if the ray pV reaches V before crossing A, otherwise V is *outside* A. Hence, the first element A of $Arcs(E)$ is set to an arc of the largest circle inscribed in E, and the first element V of $CH(I)$ is set to the first vertex which is *not before* A.

We are now ready to present the algorithm computing an optimal tour of I in E. The aim of the algorithm is to traverse the list of vertices of $CH(I)$ and localize each vertex in the planar map generated by the arcs in $Arcs(E)$ and the boundary of E (cf. Fig. 4). Once the cell containing the current vertex is determined, its critical arc may be computed easily.

Each iteration of the main step of the algorithm performs one among five possible actions. The action depends on the position of V with respect to five regions determined by the current arc A. Let $next(A)$ denote the successor of A in the list $Arcs(E)$ and let α be the smallest arc of the Voronoi circle C extending $next(A)$ and containing all the tangent points between C and E. Notice that α is completely outside A. (see Fig. 5). V falls into $\boxed{1}$, if it is outside A but not outside α, and in Region $\boxed{2}$ if it is outside α. V is in Region $\boxed{3}$ if it is inside A. Finally, V is in Region $\boxed{4}$ if it is after A, and in Region $\boxed{5}$ if it is before A.

Algorithm Optimal Tour

Input: A convex polygon E of m vertices and a simple polygon I of n vertices internal to E.

Output: A tour of I in E with the lowest possible curvature bound κ.

špace-0.1in

1. Compute CH(I). Choose a point p inside CH(I).
2. Compute SkT(E).
3. Compute $Arcs(E)$ sorted by the arcs' first endpoint around p, starting by an arc of the largest circle inscribed in E.
4. $V \leftarrow first(CH(I))$. $A \leftarrow first(Arcs(E))$. $r \leftarrow radius(A)$.
5. **while** $Arcs(E)$ is not empty and $CH(I)$ is not empty **do**

   case the region containing V **do**

   1 $r \leftarrow min(r,$ radius of critical arc of $V)$.
   $V \leftarrow next(V)$.

   2 $A \leftarrow next(A)$.

   3 $V \leftarrow next(V)$.

   4 $A \leftarrow next(A)$.

   5 $V \leftarrow next(V)$.

6. Output ζ, the maximal path internal to E, with curvature bounded by $\kappa = 1/r$.

End of the Algorithm

3.2.1 The Correctness of the Algorithm

To prove the correctness of the algorithm, we first have to show that the algorithm finds the minimum among the radii of critical circles of vertices of I. Thus, by Lemma 1, any tour of I in E would have a curvature at least as great as the curvature of that circle.

The aim of the algorithm is to locate the vertices of $C(I)$ in the planar map induced by the arcs of $Arcs(E)$ and the boundary of E. In Case 1, the extremities of arcs A and α extending $next(A)$ lie on the same two edges of E. This follows from the fact that the Voronoi circles containing A and $next(A)$ are centered on the

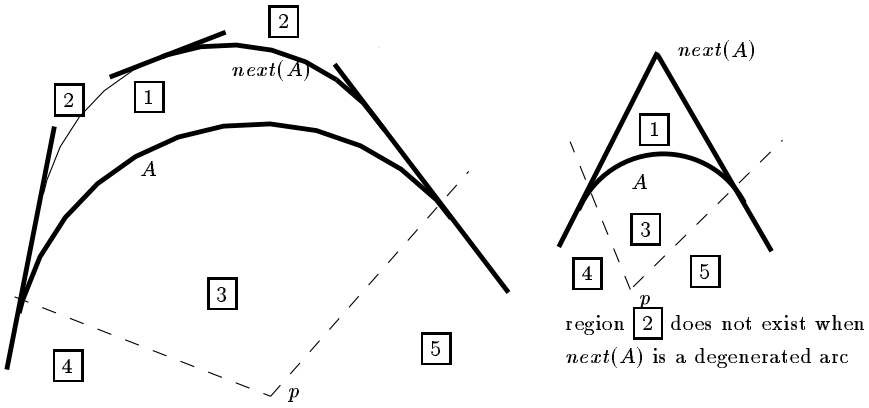

Figure 5: Illustrating algorithm Optimal Tour

same edge of SkT(E). These two edges of E define the edge of SkT(E) on which the Voronoi circles are centered. Hence, the cell of the planar map containing V is defined by two edges and two arcs. The critical arc of any point lying in that cell must be tangent to the two edges and can be computed in constant time.

Observe that in case 5 the predecessor of A in $Arcs(E)$ was an arc of radius smaller than A. Furthermore, at least one of the predecessors of V has been found outside an arc A' of radius smaller than A. It is easy to prove the same in case 3.

Thus, the algorithm finds the vertices of I whose critical arcs have the smallest radius. Then, the maximal path computed in Step 6 must be a tour of I. Otherwise, there would be a vertex of I lying outside ζ. By Lemma 1, the critical arc of that vertex would have a radius smaller that r which is impossible.

3.2.2 The Complexity of the Algorithm

The first two steps of the algorithm rely on well known optimal algorithms. The convex hull of I can be computed in $O(n)$ time, and the skeleton tree of E can be computed in $O(m)$ time. In Step 3, the list $Arcs(E)$ can be constructed in $O(m)$ time according to Lemma 3. Step 5 represents the core of the algorithm. Each iteration of the loop takes a constant time. However, as each iteration removes one vertex of CH(I) or one arc of $Arcs(E)$, the overall time complexity of this step is in

$O(n+m)$. Finally, by Lemma 2, the optimal tour ζ can be computed in $O(k)$ time, where $k \leq m$. Hence, we obtain the following result.

Theorem 4 *An optimal tour of a simple polygon with n vertices in a convex polygon with m vertices can be computed in $O(n+m)$ time and space.*

Note that the algorithm can be adapted to compute a constrained optimal tour of I in E. Suppose that the tour is constrained to be tangent to some given lines when passing through some s given points of $E \setminus I$. Let E' denote the intersection of E with s half-planes delimited by the given lines, and let I' denote the convex hull of I and the given s points. Then, we simply have to compute an optimal tour of I' in E'.

Corollary 5 *An optimal tour of a simple polygon with n vertices in a convex polygon with m vertices, constrained to have given tangents when passing through s given points, can be computed in $O(n+m+s\log s)$ time and $O(n+m+s)$ space.*

4 The Dynamic Setting

The motion planing problem considered in the previous section can be reformulated in a dynamic setting. In this case, we want to preprocess a convex polygon E with m vertices in such a way that for any given query polygon I with n vertices, we can find quickly an optimal tour of I in E.

This dynamic problem can be solved by adapting Algorithm Optimal Tour. In Step 5, if the vertex V lies in Region $\boxed{4}$ with respect to the arc A ,the list $Arcs(E)$ is processed in order but it is clear that V remains in Region $\boxed{4}$ with respect to all other arcs outside A. Those arcs correspond to the subtree of $\mathrm{SkT}(E)$ rooted at a child of the vertex on which A is centered. This subtree can be skipped in the traversal of $Arcs(E)$. Hence, the list $Arcs(E)$ is not produced explicitly in Step 3, but it may be obtained by traversing $\mathrm{SkT}(E)$ in Step 5. The subtree of $\mathrm{SkT}(E)$ effectively traversed is a subset of the subtree of $\mathrm{SkT}(E)$ used to generate an optimal tour in Step 6. Thus, the time complexity of Step 5 of the algorithm can be reduced to $O(n+k)$, where k represents the complexity of the tour. This gives us the following result.

Theorem 6 *It is possible to preprocess a convex polygon E with m vertices in $O(m)$ time and space, so that for any simple polygon I with n vertices, an optimal tour of I in E can be computed in $O(n+k)$ time, where k is the complexity of the tour.*

If the obstacle is given as a set of n points instead of a simple polygon, we simply have to compute the convex hull of these points and to appply the above result.

Corollary 7 *It is possible to preprocess a convex polygon E with m vertices in $O(m)$ time and space, so that for any set S of n points, an optimal tour of S in E can be computed in $O(n \log n + k)$ time, where k is the complexity of the tour.*

If one is just interested in computing the curvature of the optimal tour instead of computing the tour itself, an alternative solution may be used.

The main problem is still to compute the critical circle of each point of I. The discussion from the previous section shows that this problem reduces to a point location problem in the planar map induced by the arcs of $Arcs(E)$ and the boundary of E. For each vertex v of I, we locate v in the map and compute its critical arc in E. Recall that a typical cell of the map is bounded by two arcs and by two portions of edges of E. As a critical arc of any point v falling into the cell must be tangent to both edges, it may be computed in constant time once v has been located in the map.

The planar map has $O(m)$ size and it can be decomposed into trapezoids in $O(m)$ time. Following the idea of [Kir83], this decomposition can be preprocessed in $O(m)$ time and space, so that the point location would be possible in $O(\log m)$ time. The query time would now be $O(n \log m)$ to find the smallest critical circle. Hence, we obtain the following result.

Theorem 8 *It is possible to preprocess a convex polygon E with m vertices in $O(m)$ time and space, so that for any set S of n points, the curvature of an optimal tour of S in E can be computed in $O(n \log m)$ time.*

If m is much smaller than n, this method may be interesting even for computing the tour itself. The following corollary can be used alternatively to Corollary 7.

Corollary 9 *It is possible to preprocess a convex polygon E with m vertices in $O(m)$ time and space, so that for any set S of n points, an optimal tour of S in E can be computed in $O(n \log m + k)$ time, where k is the complexity of the tour.*

Observe that similar generalization may be obtained for the case of many obstacles defined as points or polygons provided they all have to be situated in the interior of the tour. In such cases, we simply have to compute the convex hull of the obstacles and then to compute an optimal tour of that new "obstacle".

5 Conclusions

The paper gives an efficient algorithm computing a smallest curvature motion of a point robot around an obstacle inside a convex polygon. The solution easily generalizes on the case of numerous obstacles. We explore the fact that the resulting path must be convex. In this case, it is sufficient to compute the curvature constraints imposed by obstacles. The maximal constraint κ is used to compute the maximal curve, internal to the workspace, which must surround all the obstacles. The idea works only in the case of convex motion, and it is not clear how it may be generalized on the case of motion admitting left and right turns.

An obvious line of further research is to design algorithms for more general workspace. From the result of [FW88] it is possible to draw a pessimistic inference that a polynomial time algorithm computing curvature-constrained motion of a point in general workspace may not exist. It is natural to ask what are more general settings, that the one studied in this paper, for which the problem of curvature-constrained motion of a point admits an efficient solution, and what are the instances of the problem which are NP-complete.

Acknowledgements

The authors would like to thank Machin for pointing out the applicability of [GT91] to this problem. They would also like to thank Jean-Pierre Merlet for supplying us with his interactive drawing preparation system JPdraw .

References

[AGSS89] A. Aggarwal, L. J. Guibas, J. Saxe, and P. W. Shor. A linear-time algorithm for computing the Voronoi diagram of a convex polygon. *Discrete Comput. Geom.*, 4:591–604, 1989.

[CDRX88] J. Canny, B. R. Donald, J. Reif, and P. Xavier. On the complexity of kinodynamic planning. In *Proc. 29th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 306–316, 1988.

[Dub57] L. E. Dubins. On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents. *Amer. J. Math.*, 79:497–516, 1957.

[FW88] S. Fortune and G. Wilfong. Planning constrained motion. In *Proc. 20th Annu. ACM Sympos. Theory Comput.*, pages 445–459, 1988.

[GT91] M. Goodrich and R. Tamassia. Dynamic trees and dynamic point location. In *Proc. 23rd Annu. ACM Sympos. Theory Comput.*, pages 523–533, 1991.

[JC89] P. Jacobs and J. Canny. Planning smooth paths for mobile robots. In *Proc. IEEE Internat. Conf. Robot. Autom.*, pages 2–7, 1989.

[Kir83] D. G. Kirkpatrick. Optimal search in planar subdivisions. *SIAM J. Comput.*, 12:28–35, 1983.

[Lat91] J.-C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, 1991.

[Ó'D87] C. Ó'Dúnlaing. Motion-planning with inertial constraints. *Algorithmica*, 2:431–475, 1987.

[Ove92] M. Overmars. A random approach to motion planning. Research Report RUU-CS-92-32, University Utrecht, Netherlands, 1992.

[Pre77] F. P. Preparata. The medial axis of a simple polygon. In *Proc. 6th Internat. Sympos. Math. Found. Comput. Sci.*, volume 53 of *Lecture Notes in Computer Science*, pages 443–450. Springer-Verlag, 1977.

[RS85] J. H. Reif and M. Sharir. Motion planning in the presence of moving obstacles. In *Proc. 26th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 144–154, 1985.

[SS90] J. T. Schwartz and M. Sharir. Algorithmic motion planning in robotics. In J. van Leeuwen, editor, *Algorithms and Complexity*, volume A of *Handbook of Theoretical Computer Science*, pages 391–430. Elsevier, Amsterdam, 1990.

[Š93] P. Švestka. A probabilistic approach to motion planning for car-like robots. Research Report RUU-CS-93-18, University Utrecht, Netherlands, 1993.

[Wil88] G. Wilfong. Motion planning for an autonomous vehicle. In *Proc. IEEE Internat. Conf. Robot. Autom.*, pages 529–533, 1988.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399