Modeling shapes with skeletons: scaffolds & anisotropic convolution

Alvaro Javier Fuentes Suárez

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES FONDAMENTALES ET APPLIQUÉES

THÈSE DE DOCTORAT

Modélisation de formes à l'aide de squelettes: échafaudages & convolution anisotrope

Alvaro Javier FUENTES SUAREZ

AROMATH, Inria Sophia-Antipolis

Présentée en vue de l'obtention du grade de docteur en Mathematiques **d'**Université Côte d'Azur

Dirigée par : Evelyne Hubert

Soutenue le : 27/09/2019

Devant le jury, composé de :
Jean-Daniel Boissonnat, Directeur de recherche, INRIA - Université Côte d'Azur
Marie-Paule Cani, Professeur, Ecole Polytechnique Paris-Saclay
Jorg Peters, Professeur, Université de Floride (Etats-Unis)
Geraldine Morin, Professeur, Université de Toulouse
Marco Livesu, Chercheur, Conseil National de la Recherche d'Italie
Adrien Bousseau, Chargé de recherche, INRIA Université Côte d'Azur
Evelyne Hubert, Directrice de recherche, INRIA - Université Côte d'Azur

ARCADES Marie Skłodowska-Curie Actions

Modélisation de formes à l'aide de squelettes: échafaudages & convolution anisotrope

Jury :

Président du jury
M. Jean-Daniel Boisonnant, Directeur de recherche, INRIA Université Côte d'Azur

Rapporteurs
Mme Marie-Paule Cani, Professeur, Ecole Polytechnique Paris-Saclay
M. Jorg Peters, Professeur, Université de Floride (Etats-Unis)

Examinateurs
Mme Géraldine Morin, Professeur, Université de Toulouse
M. Marco Livesu, Chercheur, Conseil National de la Recherche d'Italie
M. Adrien Bousseau, Chargé de recherche, INRIA Université Côte d'Azur

Directrice de thèse
Mme Evelyne Hubert, Directrice de recherche, INRIA Université Côte d'Azur

Résumé

Modélisation de formes à l'aide de squelettes: échafaudages & convolution anisotrope

Les squelettes se sont révélés être un outil efficace pour modéliser des formes complexes. Ils fournissent une base pour de nombreux processus allant de la modélisation implicite à la déformation et aux animations. Dans ce travail, nous abordons deux sujets liés à la modélisation avec squelette: les maillages quad-dominants à base de squelette et les surfaces lisses implicites générées à partir d'un squelette.

Étant donné un squelette constitué de segments de droite, nous décrivons comment obtenir un maillage quad-dominant d'une surface qui entoure étroitement le squelette et suit sa structure - l'*échafaudage*. Nous formalisons sous forme de programme linéaire sur les entiers le problème de la construction d'un échafaudage optimal minimisant le nombre total de quads sur le maillage. Nous prouvons la faisabilité du programme linéaire entier pour tout squelette. En particulier, nous pouvons générer ces échafaudages pour des squelettes avec des cycles. Nous montrons également comment obtenir des échafaudages *réguliers*, c'est-à-dire des échafaudages avec le même nombre de quads autour de chaque segment de droite, et des échafaudages *symétrique* respectant les symétries du squelette. Des applications à la polygonisation de surfaces implicites à base de squelettes sont également présentées.

Les surfaces de convolution avec des squelettes 1D ont été limitées à des sections normales presque circulaires. La nouvelle formulation que nous présentons ici augmente les possibilités de modélisation car elle permet les sections normales ellipsoïdales. Cette anisotropie est definie pour des courbes squelettales $\mathscr{G}^1$, choisies comme des splines circulaires, en interpolant l'angle de rotation et les trois rayons d'ellipsoïdes donnés, par l'utilisateur, à chaque extrémité de la courbe. Ce modèle léger crée des formes lisses qui nécessitaient auparavant de peaufiner le squelette ou de le compléter avec des pièces 2D. L'invariance par homotetie de notre formulation permet un contrôle fin des rayons et se prête ainsi à approximer une variété de formes. La construction d'un *échafaudage* est étendue aux squelettes avec des branches $\mathscr{G}^1$. Il se projette sur la surface de convolution pour former un maillage quad-dominant avec un flux d'arrêtes qui longe le squelette.

En plus des deux contributions principales décrites ci-dessus, nous développons d'autres sujets liés aux échafaudages et aux surfaces de convolution. Nous discutons la façon dont les diagrammes de Laguerre sphériques peuvent être utilisés pour améliorer la forme des échafaudages lorsque différents rayons incidents sont autorisés au niveau des articulations, et nous décrivons comment construire des maillages hexaédriques volumétriques pour un modèle basé sur un squelette à partir d'un échafaudage. Nous introduisons également les techniques de Télescopage Créatif pour le calcul par récurrence de formes closes de fonctions de convolution. Enfin, nous présentons PySkelton - une bibliothèque Python pour la modélisation basée sur le squelette qui implémente nos algorithmes et fournit une interface de programmation conviviale pour les académiques.

Mots clé: Modélisation à base de squelettes, surfaces implicites, échafaudages, surfaces de convolution

Abstract

Modeling shapes with skeletons: scaffolds & anisotropic convolution

Skeletons have proved to be a successful tool in modeling complex shapes. They provide a basis for many processes ranging from implicit modeling, to deformation and animations. In this work we advance in two topics related with skeleton modeling: quad dominant skeleton-based meshes and smooth implicit surfaces generated from a skeleton.

Given a skeleton made of line segments we describe how to obtain a coarse quad mesh of a surface that tightly encloses the skeleton and follows its structure – the *scaffold*. We formalize as an Integer Linear Program the problem of constructing an optimal scaffold that minimizes the total number of quads on the mesh. We prove the feasibility of the Integer Linear Program for any skeleton. In particular we can generate these scaffolds for skeletons with cycles. We additionally show how to obtain *regular* scaffolds, i.e. scaffolds with the same number of quad patches around each line segment, and *symmetric* scaffolds that respect the symmetries of the skeleton. Applications to polygonization of skeleton-based implicit surfaces are also presented.

Convolution surfaces with 1D skeletons have been limited to close-to-circular normal sections. The new formalism we present here increases the modeling freedom since it allows for ellipsoidal normal sections. The new anisotropy for $\mathscr{G}^1$ skeletal curves, chosen as circular splines, is interpolated from the rotation angles and three radii of ellipsoids at each extremity, given as user input. This lightweight model creates smooth shapes that previously required tweaking the skeleton or supplementing it with 2D pieces. The scale invariance of our formalism achieves excellent radii control and thus lends itself to approximate a variety of shapes. The construction of a *scaffold* is extended to skeletons with $\mathscr{G}^1$ branches. It projects onto the convolution surface as a quad mesh with skeleton bound edge-flow.

In addition to the two main contributions described above we develop further topics related to scaffolding and convolution surfaces. We discuss how spherical Laguerre diagrams may be used to improve the scaffold shapes when different incident radii is allowed at joints, and we describe how to construct volumetric hexahedral meshes for a

skeleton-based model starting from a scaffold. We also introduce Creative Telescoping techniques for the computation of closed form formulas through recurrence. Finally we present PySkelton – a Python library for skeleton based modeling that implements our algorithms and provides an academic friendly programming interface.

Keywords: Skeleton-based modeling, implicit surfaces, scaffolds, convolution surfaces

Contents

List of Figures

List of Tables

General introduction

This introductory chapter provides an overview of the contributions in this thesis. Since we kept the chapters of this thesis as close as possible to their original publications, more discussion on the state of the art can be found within the individual chapters.

Modeling shapes with skeletons has proved useful in many applications, as discussed in the very good survey in [Tagliasacchi et al., 2016]. A skeleton, made of a set of curves and/or surfaces centered inside a shape, provides a structure that encapsulates some properties of the shape while also being easier to manipulate. In this work we focus on two geometric modeling applications for 1D skeletons: the generation of a surrounding mesh and the construction of an implicit surface. There is special emphasis on the mathematical foundations of the constructions we propose.

Scaffolds. Generally speaking a *scaffold* is a mesh constructed around a skeleton made of line segments and such that it "follows" the skeleton (see chapters 1 and 2). A simple way to look at a scaffold is as a "thickening" of a 1D skeleton into a surface (see Figure 1). One can also say that a scaffold is a "cage-like" or "truss" structure.

These structures are key steps in many applications. They have been used for artistic purposes [Hart, 2002, Hart, 2008], in architecture [Srinivasan et al., 2005], for modeling [Bærentzen et al., 2012, Panotopoulou et al., 2018], sculpting modeling [Ji et al., 2010, Wu and Liu, 2012], compatible quadrangulation [Yao et al., 2009], semi-regular quad meshing [Usai et al., 2015], volumetric hexahedral meshing [Livesu et al., 2016], cage generation for posing [Casti et al., 2019], and others.

For most applications, quad-dominant scaffolds are desirable, that is scaffolds with a majority of quad patches. The main difficulties in the generation of scaffolds arise from the presence of cycles and high valency *joints* (nodes where several skeleton pieces meet). As undesired effect we get the presence of spurious patches on the scaffold that are not associated to any piece of the skeleton. Example of methods generating extra quads are those in [Yao et al., 2009] and [Usai et al., 2015]. In [Yao et al., 2009, Figure 4] one can appreciate spurious quads around joints, while in [Usai et al., 2015, Figure 6] the extra quads are due to the presence of a cycle in the skeleton. In [Ji et al., 2010] the extra patches around joints are triangular patches, while [Bærentzen et al., 2012] cannot handle skeletons with cycles.

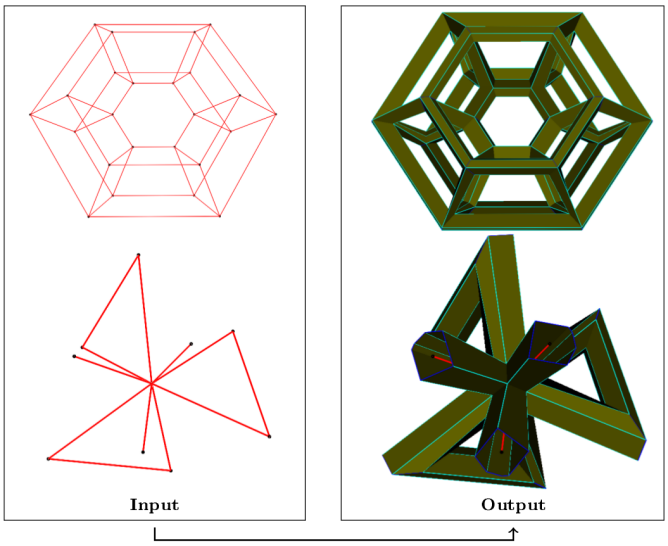
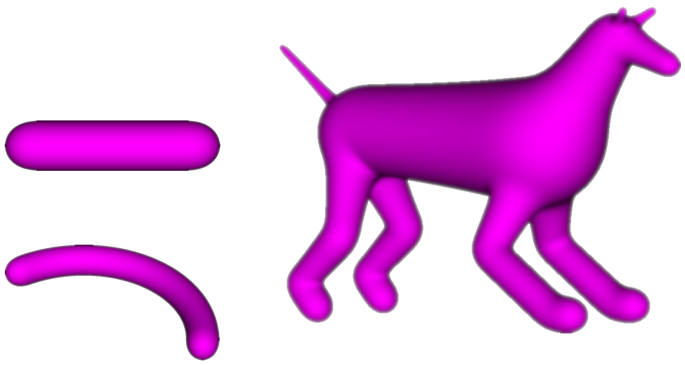


Figure 1: Scaffold construction.

In Chapter 1 we present a method for generating scaffolds that can handle skeletons with cycles. Our method does not create spurious quads around joints, and allows to construct scaffolds that satisfy any group of symmetries of the input skeleton. We construct an optimal scaffold by minimizing the total number of quads in an Integer Linear Programming model, for which we formally prove feasibility. In Chapter 2 we describe how to generate a hexahedral volumetric mesh from the scaffolds we construct. Avoiding spurious quads is not only useful for modeling and post-processes, but the total lack of spurious quads is a key property of our method that made possible the volumetric hexahedral meshing. Also in Chapter 2 we present a variant of our scaffolding method that would provide more control over the shape of the scaffolds.

Convolution surfaces. In skeleton-based modeling the user starts with a set of 1D (curves) or 2D (surfaces) objects that serves as skeleton for a shape surrounding it. Surfaces built around a skeleton are very useful since the skeleton provides an intuitive way to manipulate the surface. Several methods have been proposed for the generation of surfaces around 1D skeletons: sweep surfaces [Requicha, 1980], offset surfaces [Pham, 1992], canal surfaces [Peternell and Pottmann, 1997], B-Meshes [Ji et al., 2010], and convolution surfaces [Blinn, 1982, Bloomenthal and Shoemake, 1991, Zanni, 2013] are some of them. Convolution surfaces are specially attractive because they not only add radii information to the skeleton, but also provide a way to blend individual pieces in a smooth way (see Figure 2).

Convolution surfaces have been applied to tree modeling [Zhu et al., 2015a,



(a) Simple pieces. (b) Complex model.

Figure 2: Convolution surfaces

Zhu et al., 2015b], character modeling [Zanni et al., 2011], sketch-based modeling [Entem et al., 2015, Bernhardt et al., 2008, Zhu et al., 2011, Wither et al., 2009], immerse modeling in virtual environments [Zhu et al., 2017], and others. More specifically, a convolution surface is an implicit surface defined as a level set of a scalar field, the convolution field, that is obtained by integrating a kernel function over the skeleton. This technique can be used also to model volumes and are suitable to be combined in other implicit modeling frameworks [Pasko et al., 1995, Wyvill et al., 1999].

The mathematical smoothness of the surface obtained depends only on the smoothness of the kernel. The kernel function (power inverse, Cauchy, compact support, ...) may be selected so as to have closed form expressions for the convolution functions associated to basic skeleton elements (line segments, triangles, ...). The additivity property of integration makes the convolution function independent of the partition of the skeleton. More general skeletons are then partitioned and approximated by a set of basic elements. The convolution function for the whole skeleton is obtained by adding the convolution functions of the constitutive basic elements. See for instance [Bloomenthal and Shoemake, 1991, Cani and Hornus, 2001, Hornus et al., 2003, Jin and Tai, 2002a, Jin and Tai, 2002b, Jin et al., 2001, Sherstyuk, 1999a, Sherstyuk, 1999b, Zanni, 2013, Zanni et al., 2013].

Line segments are the most commonly used 1D basic skeleton elements. When a skeleton consists of curves with high curvature or torsion, its approximation might require a great number of line segments for the convolution surface to look as intended. Arcs of circles form a very interesting class of basic skeleton elements in the context of convolution. This was already argued in [Jin and Tai, 2002b] for planar skeleton curves. Indeed any space curve can be approximated by circular splines in a $\mathscr{G}^1$ fashion [Nutbourne and Martin, 1988, Song et al., 2009]. A lower number of basic skeleton elements are then needed to obtain an appealing convolution surface, resulting in better

visual quality at lower computational cost.

To model a wider variety of shapes it is necessary to vary the thickness around the skeleton. Several approaches have been suggested: weighted skeletons [Hubert and Cani, 2012, Jin and Tai, 2002a, Jin et al., 2001], varying radius [Hornus et al., 2003], scale invariant integral surfaces [Zanni et al., 2013], the latter two actually providing a more intrinsic formulation.

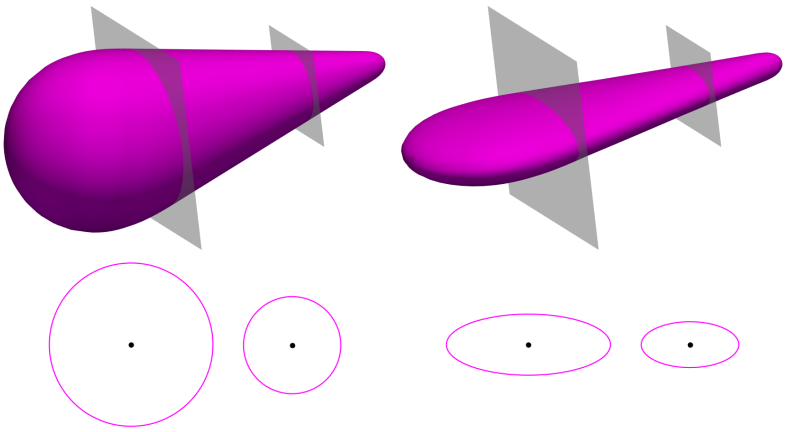
Closed form formulas, obtained through symbolic computation, have been the tool of choice for evaluating the convolution fields. While general closed form formulas were obtained for weighted line segments in [Hubert and Cani, 2012], there has been a lack of generality in terms of closed form formulas for convolution with varying radius, or scale, over line segments and, even more so, over arcs of circles. A study of the symbolic formulas for power inverse kernels is discussed in Chapter 4 with the use of some advanced computer algebra techniques.

The recurrence formulas for arcs of circle with varying radii or scale present a formidable number of terms. Their numerical evaluation can lead to instabilities. We hence feel we pushed the symbolic approach to its limits. For a more versatile approach we call on numerical integration using the well established quadrature methods implemented in QUADPACK [Piessens et al., 1983].

Anisotropy. Standard convolution surfaces around 1D skeletons have circular cross sections. That is also the case for varying radius and scale integral surfaces. This significantly restricts the shapes that can be generated. In Chapter 5 we discuss previous approaches for mitigating this issue and we introduce a new technique to generate *anisotropic* convolution surfaces. A simple and intuitive approach to control the shape is also provided. In Figure 3 we illustrate anisotropy in the context of convolution surfaces.

Medial axis, the most intrinsic skeleton, has 2D and 1D elements. Convolution surfaces around 2D skeletons have been approached in the literature [McCormack and Sherstyuk, 1998, Jin et al., 2008, Hubert, 2012, Zhu et al., 2011, Zanni et al., 2013, Zhu et al., 2017]. While useful, 2D skeletons increase the complexity in the formulas involved as well as on the modeling process for the user. In this thesis we favor the use of 1D skeletons. Anisotropic convolution limits the need of 2D parts by extending the capabilities of convolution around 1D skeletons such that one can mimic the effects of 2D skeletons, as discussed in Chapter 5.

In this work we assume that the modeled surface is topologically consistent with the skeleton. The precise radii control introduced in Chapter 5 allows the user to detect when this is not the case. Consistency between the surface and the skeleton allows to identify a good edge-flow on a surface mesh: when the edges "follow" the 1D skeleton. We explore in Chapter 5, after a brief discussion at the end of Chapter 1, the use of our scaffolds for the meshing of skeleton-consistent implicit surfaces with good edge-flow on the final mesh. For readers interested in a detailed discussion of skeleton-surface topological



(a) Standard convolution. (b) Anisotropic convolution.

Figure 3: Anisotropy in convolution surfaces. In the top row we show the surface and two slicing planes. In the bottom we show the cross sections given by the slicing planes.

consistency in the context of convolution surfaces we refer to [Ma and Crawford, 2008].

PySkelton. Our research results were implemented into a Python library. Our library makes use of established libraries for mathematical programming and numerical analysis: Qhull [Barber et al., 1996] for convex hull computations, GSL [Galassi et al., 2017] and QUADPACK for numerical integration and root finding, and GLPK [Makhorin, 2016] for integer linear programming.

Much of the content of this work has been published and presented in the following papers and venues:

Scaffolding:

- [Fuentes Suárez and Hubert, 2018b] Fuentes Suárez, A. J. and Hubert, E. (2018b). Scaffolding skeletons using spherical Voronoi diagrams: Feasibility, regularity and symmetry. Computer-Aided Design, 102:83–93.

  Presented in SPM2018: Solid and Physical Modeling. June 11-13, 2018, Bilbao, Spain.

  Topic: full development of the scaffolding algorithm, generation of optimal quad-dominant meshes that respect the symmetries of the skeleton, and with the same number of patches around each line segment (*regularity*).

  Full content in Chapter 1.

- [Fuentes Suárez and Hubert, 2017] Fuentes Suárez, A. J. and Hubert, E. (2017). Scaffolding skeletons using spherical Voronoi diagrams. Electronic Notes in Discrete Mathematics, 62:45–50.

  Presented in LAGOS2017: IX Latin and American Algorithms, Graphs, and Optimization Symposium. September 11-15, 2017, CIRM, Marseille, France.

  Topic: proof of feasibility for the construction of a scaffold (quad dominant mesh that follows the structure of the skeleton) for skeletons made of line segments of any topology.

  Extended abstract.

- Poster presentation: *Scaffolding skeletons using spherical Voronoi diagrams*. FoCM2017: Foundations of Computational Mathematics. July 10-19, 2017, Barcelona, Spain.

  Convolution surfaces:

- [Fuentes Suárez et al., 2019] Fuentes Suárez, A. J., Hubert, E., and Zanni, C. (2019). Anisotropic convolution surfaces. Computers & Graphics, 82:106–116.

  Presented in SMI2019/IGS2019: Shape Modeling International 2019, International Geometry Summit 2019. June 17-22, 2019, Vancouver, Canada.

  Topic: An extension to the convolution surfaces technique that increases the modeling freedom. A scaffold-based meshing technique for implicit surfaces around a skeleton is also discussed.

  Full content in Chapter 5.

- [Fuentes Suárez and Hubert, 2018a] Fuentes Suárez, A. J. and Hubert, E. (2018a). Convolution surfaces with varying radius: Formulae for skeletons made of arcs of circles and line segments. In Research in Shape Analysis: WiSH2, Sirince, Turkey, AWM, pages 37–60. Springer.

  Topic: study of symbolic formulas for convolution surface fields, introduction of Creative Telescoping in the context of closed form formulas for convolution surfaces.

  Partial and adapted content in chapters 3 and 4.

- Poster presentation: *Anisotropic convolution for modeling 3D smooth shapes around 1D skeletons*. 9th International Conference on Curves and Surfaces. June 28-July 4, 2018, Arcachon, France.

As a summary, the contributions in this thesis are:

- Scaffolding algorithm and extensions.
- Anisotropic convolution surfaces.
- Meshing of implicit surfaces using scaffolding.
- Python library implementing scaffolding and anisotropic convolution.
- Symbolic integral formulas for convolution surfaces through recurrence.

This manuscript is structured as follows. In Chapter 1 we introduce our scaffolding method. In Chapter 2 we discuss potential extensions and generalizations of scaffolding. In Chapter 3 we introduce convolution surfaces. In Chapter 4 we present the closed form formulas for convolution fields. Chapter 5 is devoted to our anisotropic extension of convolution surfaces. Finally in Chapter 6 we present the library we developed with the implementations of the methods and algorithms from chapters 1 and 5. We conclude with some general overviews and comments on future development.

Part I

Scaffolds

Chapter 1

Scaffolding skeletons using spherical Voronoi diagrams: feasibility, regularity and symmetry

This chapter was published in:

[Fuentes Suárez and Hubert, 2018b] Fuentes Suárez, A. J. and Hubert, E. (2018b). Scaffolding skeletons using spherical Voronoi diagrams: Feasibility, regularity and symmetry. Computer-Aided Design, 102:83–93.

An extended abstract version was also published in:

[Fuentes Suárez and Hubert, 2017] Fuentes Suárez, A. J. and Hubert, E. (2017). Scaffolding skeletons using spherical Voronoi diagrams. Electronic Notes in Discrete Mathematics, 62:45–50.

1.1 Introduction

Skeletons are used in 3D graphics for modeling and animating articulated shapes. The user can design a complex shape by sketching a simple geometric object that is the input to a surface generating algorithm. By making changes in the skeleton it is possible to change the shape in an intuitive way. Due to their low dimensional nature, skeletons can serve as an efficient and compact representation of a surface. In this context a skeleton-based mesh generation method is needed.

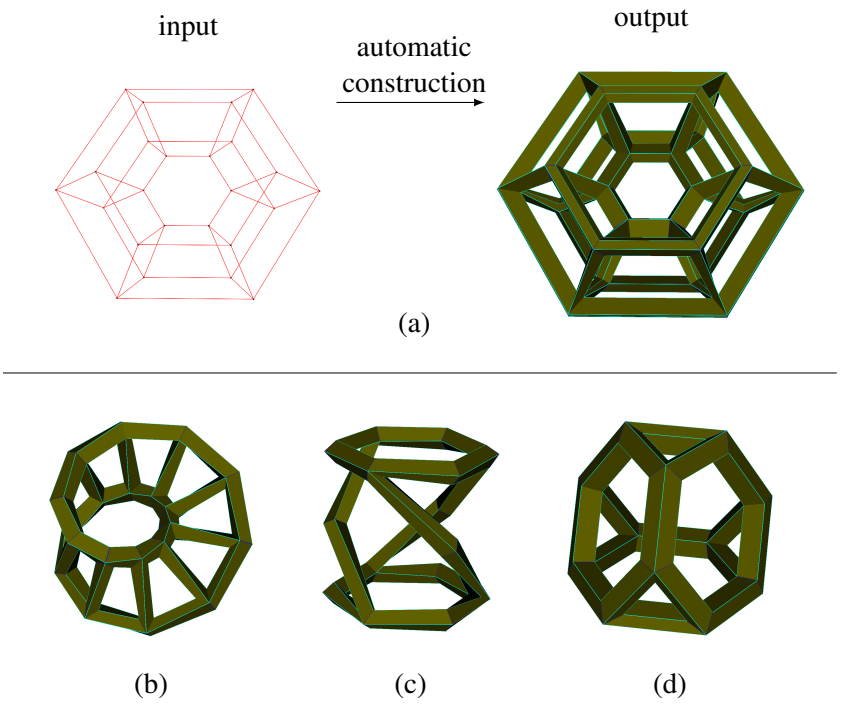


Figure 1.1: Recreation of some of the scaffolds used in [Karčiauskas and Peters, 2016], automatically computed by our method from suitable skeletons.

The idea developed here is to construct a "coarse" quad mesh that tightly follows the structure of the skeleton. Following the terminology of [Panotopoulou et al., 2018] we call this process *scaffolding* and the corresponding coarse mesh a *scaffold*. This is used as an *intermediate* step in many applications. A scaffold can be used to generate a surface, either by subdivision as in [Bærentzen et al., 2012], or as initial patchwork for a spline surface [Blidia et al., 2017, Karčiauskas and Peters, 2016] (see Figure 1.1). It is used as an intermediate step in the extraction of a quad layout on a given triangular mesh [Bærentzen et al., 2014, Usai et al., 2015], and for compatible quadrangulation [Yao et al., 2009]. An application we present here is the polygonization of skeleton-based implicit surfaces into quad-dominant meshes. In particular, we use our method for visualization of convolution surfaces [Bloomenthal and Shoemake, 1991, Zanni, 2013].

Our main contribution is in the theoretical foundations of an algorithm that computes a scaffold for any skeleton, independently of its topology, and with no user interaction.

In this paper we deal with skeletons made of line segments that do not intersect except at the endpoints, then called *joints*.

1.1.1 Previous work

One of the earliest ideas for a scaffold construction was to sweep a fixed polygonal cross profile along the segments, stitching the generated quads by means of a convex hull construction at the joints. This idea was first used in [Srinivasan et al., 2005], producing meshes with some triangular faces resulting from the stitching process. For quadrilateral cross profiles B-meshes method [Ji et al., 2010] improved upon this by merging triangles in order to get a quad-dominant mesh. Still the stitching process might leave some triangular patches at the joints.

[Usai et al., 2015] and [Yao et al., 2009] proposed a scaffolding technique that generates a pure-quad mesh by extruding boxes emanating from cubes positioned at the joints. In [Usai et al., 2015] the subdivision of the cubes is modeled with an Integer Linear Program, for which a solution might fail to exist when there are cycles in the skeleton. "Lids" [Usai et al., 2015, Figure 6] are introduced as a workaround. [Yao et al., 2009] also introduces extra quads around joints [Yao et al., 2009, Figure 4].

The method we propose is based on the Skeleton to Quad-dominant polygonal Mesh (SQM) method in [Bærentzen et al., 2012] which was limited to skeletons without cycles. SQM first defines the mesh points around the joints and then recreate a quad based "tubular" polyhedral surface around each line segment. SQM can be regarded as a three steps process:

1. **Partition** the unit sphere centered at the joints into regions, one for each incident line segment.

2. **Discretize** each region into a cell (as an ordered set of points on its boundary) such that the two cells at the extremities of each line segment are *compatible*, i.e. have equal number of points. Points on the boundary of two regions are part of the two corresponding cells.

3. **Link**, in a bijective way, the cells at the extremities of each line segment. These links define the quads on the mesh.

For Step 3, SQM [Bærentzen et al., 2012] defines the links by minimizing the total length of the line segments they define. In Step 2 Bærentzen *et al.* propose an algorithm for inserting additional vertices on the cells in such a way that the compatibility constraint is satisfied. Yet this algorithm does not work in the presence of cycles [Bærentzen et al., 2012]. The existence of a possible discretization was actually

not proved. Furthermore there was no analysis on the optimality of SQM with respect to the number of quads in the scaffold.

The partition of the sphere, in Step 1, used in [Bærentzen et al., 2012] can be recognized to be a Voronoi diagram on the sphere. [Panotopoulou et al., 2018] introduces a partition of the sphere in quadrangles that makes the compatibility of cells (Step 2) trivial. Yet the partition is not canonical and the convexity of the regions is not guaranteed.

1.1.2 Contributions

Our method follows SQM [Bærentzen et al., 2012] but we address and solve the crux difficulty that represents Step 2 for skeletons of arbitrary topology. We formalize the creation of compatible cells (Step 2) as an Integer Linear Program (IP) that minimizes the total number of quads. We prove that, for a Voronoi partition of the spheres, there exists a solution for the IP (feasibility), even in the presence of cycles. There is thus always an optimal solution that can be computed by an IP solver. Feasibility is proven thanks to a numerical characterization by Rivin [Rivin, 1996] of graphs combinatorially equivalent to inscribable polyhedrons (i.e. with vertices on a sphere) that applies to the dual of the Voronoi diagram. Our minimization criterion is what ensures the coarsest mesh among those based on Voronoi partition of the spheres and additional pragmatic geometric restrictions. The solution of the IP determines the cross profile on each segment.

We present two other constructions to generate *symmetric* and *regular* scaffolds. In the former case the scaffold respects the symmetries of the skeleton. In the latter case, the scaffold has equal number of quads around each line segment of the skeleton, ensuring a similar cross profile for each line segment. Both possibilities are natural requirements for geometric modeling. With either or both requirements, we prove the feasibility of the constraints and are thus in a position to compute optimal solutions in the total number of quads.

To the extent of the knowledge of the authors the only paper that integrates the symmetries of the skeleton into the computation of the scaffold is [Bærentzen et al., 2012], with the limitation that only one reflection symmetry can be taken into account for each skeleton (in addition to being restricted to cycle-free skeletons). Here we present a much more general approach that is able to compute scaffolds that respect any group of symmetries of the skeleton.

The article is organized as follows. In Section 1.2 we formalize the notions of skeleton and scaffold. In Section 1.3 we prove the existence of: standard, regular, and symmetric scaffolds, for any topology. In Section 1.4 we define an objective function and introduce the IPs that find optimal solutions. The algorithms to construct a scaffold are detailed in Section 1.5 with some further discussion in Section 1.6. An application to polygonization of skeleton-based implicit surfaces is shortly presented in Section 1.7.

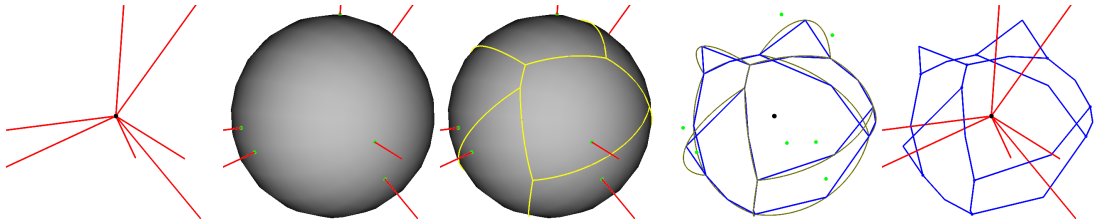
A sketch of the first feasibility proof was presented at the conference LAGOS 2017. An extended abstract of the talk given there is available in the proceed-

ings [Fuentes Suárez and Hubert, 2017]. Here we generalize and extend the result in [Fuentes Suárez and Hubert, 2017]. The precise objective function of the IP, the details of the algorithms, as well as the regular and symmetric cases are presented in this paper for the very first time.

1.2 Skeletons & Scaffolds

In this paper a *skeleton* is a finite set Σ of spatial line segments satisfying the following property: any two line segments intersect at most at one of their endpoints. A skeleton Σ defines naturally a graph $G_\Sigma = (\mathscr{V}_\Sigma, \mathscr{E}_\Sigma)$ by identifying the set of nodes $\mathscr{V}_\Sigma$ with the set of all endpoints in Σ, and the set of edges $\mathscr{E}_\Sigma$ with the set Σ itself. An edge $e \in \mathscr{E}_\Sigma$ connecting two nodes $a, b \in \mathscr{V}_\Sigma$ can be alternatively represented as ab. If $e = ab$ we say that e is *incident* to a (or b) and write $e \multimap a$ (or $e \multimap b$). A node with only one incident edge is called *dangling* node while a node with more than one incident edge is called a *joint*. A node with precisely two incident edges is called an *articulation*. We denote by $L_\Sigma \subset \mathscr{V}_\Sigma$ the set of all dangling nodes, $M_\Sigma \subset \mathscr{V}_\Sigma$ the set of all articulations, and $N_\Sigma = \mathscr{V}_\Sigma - (L_\Sigma \cup M_\Sigma)$ the set of the remaining joints.

The sphere centered at $v \in \mathscr{V}_\Sigma$ with radius $\varepsilon_v > 0$ is denoted $\mathbb{S}_v$, and $\mathscr{A}_v = \{e \cap \mathbb{S}_v \mid e \in \mathscr{E}_\Sigma, e \multimap v\}$ is the set of the points that are the intersection of the line segments incident to v with $\mathbb{S}_v$ (Figure 1.2b). For what follows the choices of ε_v ($v \in \mathscr{V}_\Sigma$) is independent of our method. We assume though that no two spheres intersect. Note that, once the scaffold mesh is created, $\max_{v \in \mathscr{V}_\Sigma} \varepsilon_v$ is an upper bound to the distance between any point in the edges of the scaffold and the skeleton.



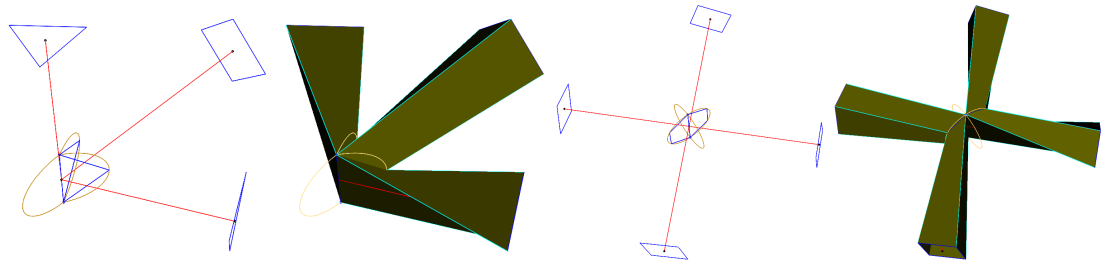
(a) (b) (c) (d) (e)

Figure 1.2: Construction of cells: for every joint (a) take the intersection of the incident segments with the unit sphere (b), compute the Voronoi diagram (c), subdivide the arcs in the boundaries of Voronoi regions (d) and take the ordered set of points (polyline) as representation of the cells (e).

For a joint $v \in \mathscr{V}_\Sigma$, the *Voronoi diagram* [Aurenhammer, 1991] of $\mathscr{A}_v$ on $\mathbb{S}_v$ (Figure 1.2c), denoted $\mathrm{Vor}(\mathscr{A}_v)$, partitions the sphere $\mathbb{S}_v$ into regions $\{R^v_e\}_{e \multimap v}$, with $(\mathbb{S}_v \cap e) \in R^v_e$, that are delimited by arcs of great circles [Augenbaum and Peskin, 1985, Na et al., 2002]. The cell C^v_e associated to the region R^v_e consists of the end-points of the arcs delimiting R^v_e and some additional points (possibly none) per arc (Figure 1.2d).

The points chosen in one arc define a polyline that represents the arc (Figure 1.2e). The number of segments in the polyline is called *number of subdivisions* of the arc, it is one less than the number of points taken in the arc.

One should also consider some geometrical constraints. The number of points in a cell must be at least 3 (4 is more customary [Ji et al., 2010, Panotopoulou et al., 2018, Usai et al., 2015, Yao et al., 2009]). *Long arcs*, i.e. arcs with length greater than or close to π, must be subdivided into at least two segments. Examples of degenerate cases for cells with at least 3 or 4 points are shown in Figure 1.3.



(a) (b)

Figure 1.3: Degenerate cases arise when allowing for only one subdivision per arc, with at least 3 (a), or 4 (b), points on each cell. In both cases *long* pole-to-pole arcs are discretized as single segments going through the joint.

If the node v is an articulation, the arc separating the two Voronoi regions in $\mathscr{A}_v$ is a circle. Thus an associated cell C_e^v consists of points on the boundary circle. For v a dangling node, $\mathrm{Vor}(\mathscr{A}_v)$ consists of a single region with no boundary. In this case the associated cell C_e^v consists of points on the circle defined by the sphere $\mathbb{S}_v$ and the plane through v normal to e. In both cases the points on the cells are given by only one arc (great circle) and are taken such that the planar polygon they define encloses v.

There seems to be a preference in the literature for at least quadrangular cross profiles [Ji et al., 2010, Panotopoulou et al., 2018, Usai et al., 2015, Yao et al., 2009]. To guarantee this the cells must have at least four points. We show in Figure 1.14 that three is also an adequate choice. In general, arcs can be discretized as a single segment (i.e. with no additional point) but the extra restrictions on the minimum number of points on each cell must be enforced to avoid cases like in Figure 1.3.

A *scaffold* $\mathscr{K}_\Sigma$ is defined as a pair (P_Σ, Φ_Σ), satisfying

1. $P_\Sigma = \{\mathscr{C}_v \mid v \in \mathscr{V}_\Sigma\}$, where each $\mathscr{C}_v = \{C_e^v \mid e \in \mathscr{E}_\Sigma, e \multimap v\}$ is a family of *cells* representing a partition of $\mathbb{S}_v$ according to $\mathrm{Vor}(\mathscr{A}_v)$.

2. $\Phi_\Sigma = \{\phi_e \mid e \in \mathscr{E}_\Sigma\}$ is a family of bijections ϕ_e between C_e^a and C_e^b for $e = ab$.

For $e = ab$ we say that C_e^a and C_e^b are *linked cells*. Similarly, if $C_e^a = \langle p_1, p_2, \ldots, p_n \rangle$ we say that p_i is *linked* with $\phi_e(p_i)$ and the pair $\langle p_i, \phi_e(p_i) \rangle$ is called a *link*. The

realization of a scaffold as a mesh is through the quads defined by the four-points tuples $\langle p_i, \phi_e(p_i), \phi_e(p_{i+1}), p_{i+1}\rangle$.

To construct the links we follow the same strategy as in [Bærentzen et al., 2012]: provided the cells have the same number of points, choose the bijection where the total length of the segments defined by the links is minimal.

The existence of the scaffold, for a given skeleton, is thus established if and only if we can discretize the Voronoi regions into compatible cells: we need the cells C_e^a and C_e^b (for all $e = ab \in \mathscr{E}_\Sigma$) to have the same number of points. This is far from obvious, specially in the presence of cycles in the skeleton.

1.3 Existence of scaffolds

We construct the set of linear equations over the integers that preside over the existence of a scaffold. The achievement is to prove the existence of a positive solution to this system, hence proving the existence of a scaffold for any given skeleton. The proof strongly relies on a property of Voronoi diagrams on the sphere, and more precisely on their duals. We examine this property first. We then prove the existence of a scaffold, which we qualify as *standard*. In geometric modeling it is desirable to have scaffolds that respect the symmetries of the underlying skeleton (*symmetric scaffold*), or to require a regularity on the number of quads around the line segments (*regular scaffold*). We can even seek scaffolds that satisfy both properties. We prove the existence of all these scaffolds.

1.3.1 Locally uniform discretization

The *Delaunay triangulation* $\mathrm{Del}(\mathscr{A}_v)$ is the dual of $\mathrm{Vor}(\mathscr{A}_v)$ [Aurenhammer, 1991]. For each $v \in \mathscr{V}_\Sigma$ let E_v be the set of edges of $\mathrm{Del}(\mathscr{A}_v)$. Each edge in E_v represents a common boundary between two regions in $\mathrm{Vor}(\mathscr{A}_v)$. For $f \in E_v$ we define a positive integer x_f^v representing the number of subdivisions to be done to the corresponding arc (i.e. the number of segments in the polyline representation of the arc). For dangling nodes $E_v = \emptyset$, we nonetheless introduce a phantom edge ν_v, with an associated variable $x_{\nu_v}^v$, so that actually $E_v = \{\nu_v\}$.

The following lemma asserts that the Voronoi regions can be discretized uniformly, i.e. with an equal number of points. It is our main ingredient in proving the existence of scaffolds.

Lemma 1.3.1. *For $v \in \mathscr{V}_\Sigma$, the* local *linear system*

$$\sum_{\substack{f \in E_v \\ f \multimap (\mathbb{S}_v \cap e)}} x_f^v = \lambda_v \quad \forall e \multimap v, e \in \mathscr{E}_\Sigma \tag{1.3.1}$$

has a solution $(\tilde{x}_f^v, \tilde{\lambda}_v)$ *with positive integer entries.*

We denote the solution $(\tilde{x}_f^v, \tilde{\lambda}_v)$ as *local solution* associated to the *local* system (1.3.1).

To guarantee a positive solution we rely in a proposition due to Rivin [Rivin, 1996] (statement extracted from [Dillencourt and Smith, 1996]) that gives a numerical characterization for a graph of inscribable type (i.e. a graph combinatorially equivalent to a polyhedron inscribed on a sphere [Dillencourt and Smith, 1996]).

Proposition 1.3.1 (I. Rivin)**.** *If a graph is of inscribable type then weights w can be assigned to its edges such that:*

i For each edge e, $0 < w(e) < 1/2$.

ii For each vertex v, the total weight of all edges incident to v is equal to 1.

Proposition 1.3.1 applies to $\mathrm{Del}(\mathscr{A}_v)$ that is combinatorially equivalent to the convex hull of $\mathscr{A}_v$ [Brown, 1979, Grima and Márquez, 2001], and hence of inscribable type. It thus guarantees a positive real solution for (1.3.1). Note that guaranteeing a positive solution for a linear system is not a trivial task.

The following claim then relates the existence of an integer positive solution of a homogeneous linear system to the existence of a real positive solution.

Proposition 1.3.2. *A homogeneous linear system with integer coefficients has a positive integer solution whenever it has a positive real solution.*

of Proposition 1.3.2. Let $Ay = 0$ be a homogeneous linear system where A is a $n \times m$ matrix with integer entries. Observe that if there is a rational solution p with $p \in \mathbb{Q}^m$, we can get an integer positive solution multiplying p by the least common multiple of denominators in the entries of p. Thus it is enough to prove that the system has a rational positive solution.

Let $\tilde{y} \in \mathbb{R}^m$ be the real solution of the homogeneous system with positive entries, this implies that the set of solutions of the system is a (non-trivial) subspace. Since A has integer entries we get a rational basis for the solution space. Let $\{y_1, y_2, \ldots, y_k\} \subset \mathbb{Q}^m$ ($k \geq 1$) be a basis of the solution space of A. We have that $\tilde{y} = \sum_{i=1}^{k} \tilde{c}_i y_i$ for some real coefficients $\tilde{c}_i$. Let $f : \mathbb{R}^k \to \mathbb{R}^m$ be a function mapping $c \in \mathbb{R}^k$ to $f(c) = \sum_{i=1}^{k} y_i c_i$. Let $U = (0,\infty)^m$. Clearly U is open, and f is continuous, thus $V = f^{-1}(U) \subset \mathbb{R}^k$ is open. Moreover $\tilde{c} = (\tilde{c}_1, \ldots, \tilde{c}_k) \in V$ hence V is not empty. The set of rational points in $\mathbb{R}^k$ is dense, thus there exists $q \in \mathbb{Q}^k \cap V$. On the other hand $f(q) \in U$, that is all the entries of $f(q)$ are positive and rational and by definition $f(q)$ is in the solution space of A. Therefore $f(q)$ is a rational solution with positives entries for the system. □

of Lemma 1.3.1. For v a dangling node or articulation, a solution is trivially found. If v is a joint with at least three incident edges, the existence of a real positive solution for the

local system (1.3.1) comes from the fact that $\mathrm{Del}(\mathscr{A}_v)$ is combinatorially equivalent to the convex hull of $\mathscr{A}_v$ [Brown, 1979, Grima and Márquez, 2001] which is an inscribed polyhedron. Thus Proposition 1.3.1 guarantees the existence of a real positive solution given by $x_f^v = w(f)$ and $\lambda_v = 1$, the result follows from Proposition 1.3.2. □

1.3.2 Standard scaffold

The number of points in a cell C_e^v ($e \in \mathscr{E}_\Sigma$ and $e \multimap v$) is given by

$$|C_e^v| = \sum_{\substack{f \in E_v \\ f \multimap (\mathbb{S}_v \cap e)}} x_f^v. \tag{1.3.2}$$

For each edge $e = ab \in \mathscr{E}_\Sigma$, there is a bijection $\phi_e \in \Phi_\Sigma$ in the scaffold $\mathscr{K}_\Sigma$ between the cells C_e^a and C_e^b. This is possible only if both cells have the same number of points, which gives the following compatibility equations

$$\sum_{\substack{h \in E_a \\ h \multimap (\mathbb{S}_a \cap e)}} x_h^a = \sum_{\substack{g \in E_b \\ g \multimap (\mathbb{S}_b \cap e)}} x_g^b \qquad \forall e = ab \in \mathscr{E}_\Sigma. \tag{1.3.3}$$

By definition x_f^v is a positive integer for all $v \in \mathscr{V}_\Sigma$, $f \in E_v$. In Section 1.2 we discussed some additional geometric constraints. They can be written in the form

$$\begin{cases} x_f^v \in \mathbb{Z}, x_f^v \geq 1 & \forall v \in \mathscr{V}_\Sigma, f \in E_v \\ \Lambda_i(x_f^v) \geq s_i & i = 1, 2, \ldots \end{cases} \tag{1.3.4}$$

where $\Lambda_i(x_f^v)$ are linear forms on the variables x_f^v with non-negative integer coefficients (not all zeros), and $s_i > 0$ are integer constants. These can capture the requirements of having at least 3 (or 4) points on each cell, as well as subdividing *long* arcs into at least two segments. The existence proof works for any set of (Λ_i, s_i) with non-negative coefficients. A practical realization of (1.3.4) for the geometric constraints commented in Section 1.2 and that should also serve as reference for the reader, is given by

$$\begin{cases} x_f^v \in \mathbb{Z}, x_f^v \geq m(x_f^v) & \forall v \in \mathscr{V}_\Sigma, f \in E_v \\ \sum\limits_{\substack{f \in E_v \\ f \multimap (\mathbb{S}_v \cap e)}} x_f^v \geq c & \forall v \in \mathscr{V}_\Sigma, e \in \mathscr{E}_\Sigma, e \multimap v \end{cases} \tag{1.3.5}$$

where c is 4 or 3; $m(x_f^v)$ is 1 if the length of the arc associated to x_f^v is less than $\pi - \delta$, and 2 otherwise. The choice of a small constant $\delta \in (0, \pi)$ determines the *long* arcs. Yet other constraints that may arise in applications, like subdividing specific arcs into a greater number of pieces, or requiring specific cells to have a greater number of points, can also be modeled in (1.3.4).

We say that the *global* system defined by (1.3.3) is *feasible* if it has a solution satisfying (1.3.4). Such a solution gives a way to discretize each region in the partition of the spheres into compatible cells, and thus allows to construct a scaffold. Proving the existence of a positive (integer or real) solution for a linear system is not a trivial task. The main result in our paper is the formal proof of feasibility of (1.3.3) which is stated in the following theorem.

Theorem 1. *For any skeleton* Σ*, the linear system given by* (1.3.3) *has a solution with the entries* x_f^v $(v \in \mathscr{V}_\Sigma, f \in E_v)$ *satisfying the constraints given in* (1.3.4)*. Therefore there exists a scaffold for* Σ*.*

Proof. Using Lemma 1.3.1, for every $v \in \mathscr{V}_\Sigma$, we take $(\tilde{x}_f^v, \tilde{\lambda}_v)$ as a *local solution* satisfying (1.3.1). Then we take $\hat{x}_f^v = s\frac{\hat{\lambda}}{\tilde{\lambda}_v}\tilde{x}_f^v$ for all $v \in \mathscr{V}_\Sigma, f \in E_v$, where $\hat{\lambda} = \prod_{u\in\mathscr{V}_\Sigma}\tilde{\lambda}_u$ and $s = \max_i s_i$ is an integer constant. Using $\hat{x}_f^v$ as subdivisions for the arcs we get that all the cells have the same number of points: $s\hat{\lambda}$, it follows then that the equalities in (1.3.3) are trivially satisfied. The factors $s\hat{\lambda}/\tilde{\lambda}_v$ guarantee that the constraints in (1.3.4) are also satisfied. Thus $\hat{x}_f^v$ is a solution to (1.3.3) satisfying (1.3.4). □

The discretization constructed in the proof above is far from optimal. Optimality is dealt with in Section 1.4 with the help of Integer Linear Programming.

1.3.3 Symmetric scaffold

It is a desirable property for a scaffold to respect the symmetries of the underlying skeleton [Bærentzen et al., 2012]. As illustrated in Figure 1.4 and 1.6, a standard scaffold need not satisfy this property. In this section we first define what is a valid symmetry for the skeleton. We then give the additional restrictions needed in order to obtain a scaffold that respects the symmetries of the skeleton.

A *skeleton symmetry* of Σ is an isometry $T : \mathbb{R}^3 \to \mathbb{R}^3$ such that

1. $T(v) \in \mathscr{V}_\Sigma \quad \forall v \in \mathscr{V}_\Sigma$, and
2. $T(e) \in \mathscr{E}_\Sigma \quad \forall e \in \mathscr{E}_\Sigma$.

If the radii ε_v are different then in order to have a symmetric scaffold they must satisfy $\varepsilon_v = \varepsilon_{T(v)}$ for all $v \in \mathscr{V}_\Sigma, T \in \mathscr{T}_\Sigma$. We assume this is guaranteed for symmetric skeletons.

Conditions (1) and (2) say that T keeps the set of nodes $\mathscr{V}_\Sigma$ and edges $\mathscr{E}_\Sigma$ (hence G_Σ) invariant, thus the whole skeleton is kept fixed: $T(\Sigma) = \Sigma$. Since T is an isometry and $e = ab \in \mathscr{E}_\Sigma$ is the line segment between the nodes a and b (including both), then $T(e)$ is the edge (line segment) connecting $T(a)$ and $T(b)$.

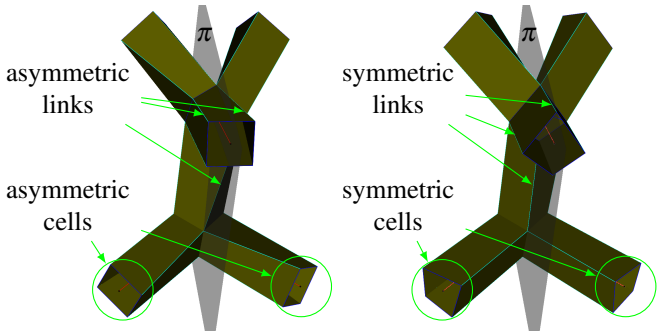


Figure 1.4: A symmetric scaffold is expected for a symmetric skeleton. In the picture the skeleton is symmetric through the plane π. (a) The cells of one standard scaffold does not respect the symmetry, links are not symmetric either. (b) The cells of a symmetric scaffold respect the symmetry.

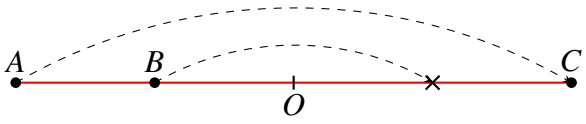


Figure 1.5: A skeleton with a false symmetry: the central symmetry with respect to O, maps the skeleton to itself when considered as a curve but not as a graph (B is not mapped to another node).

With conditions (1) and (2) we avoid cases, as illustrated in Figure 1.5, where there is a geometric symmetry for Σ that does not map elements of G_Σ into elements of G_Σ. Thus not all symmetries are *skeleton symmetries*.

The set of skeleton symmetries $\mathscr{T}_\Sigma$ forms a finite group under composition. For simplicity we denote $T \circ R$ as TR.

We say then that a scaffold $\mathscr{K}_\Sigma = (P_\Sigma, \Phi_\Sigma)$ respects the skeleton symmetry $T \in \mathscr{T}_\Sigma$ if

$$C^{T(v)}_{T(e)} = T(C^v_e) \quad \forall v \in \mathscr{V}_\Sigma, e \in \mathscr{E}_\Sigma, e \multimap v, \tag{1.3.6}$$

and

$$\phi_{T(e)} = T \circ \phi_e \circ T^{-1} \quad \forall e \in \mathscr{E}_\Sigma. \tag{1.3.7}$$

Since Voronoi diagrams depend only on the distance between points, it follows that

$$\mathrm{Vor}(\mathscr{A}_{T(v)}) = T(\mathrm{Vor}(\mathscr{A}_v)). \tag{1.3.8}$$

Notice that it is also true that $\mathscr{A}_{T(v)} = T(\mathscr{A}_v)$ and $E_{T(v)} = T(E_v)$. Hence Equation (1.3.6) can be achieved as soon as the subdivisions are done in a symmetric way for symmetric arcs. This is possible if

$$x^v_f = x^{T(v)}_{T(f)} \qquad \forall T \in \mathscr{T}_\Sigma, v \in \mathscr{V}_\Sigma, f \in E_v. \tag{1.3.9}$$

To guarantee (1.3.6) given (1.3.9), it is sufficient to subdivide the arcs into equal length subdivisions since arc-length is preserved under isometries: the arc corresponding to $f \in E_v$ ($v \in \mathscr{V}_\Sigma$) with arc-length θ is subdivided into x^v_f sub-arcs of length θ / x^v_f. Equation (1.3.7) means that the links defined by the bijections in Φ_Σ define symmetric line segments, and hence symmetric quads.

Solutions of (1.3.3) satisfying (1.3.4) and the extra constraints given by (1.3.9), yield compatible cells that respect the skeleton symmetries. The existence of such a solution is denoted as feasibility of the symmetric scaffold, and it is established in Theorem 2.

The proof of Theorem 2 relies on the following lemma.

Lemma 1.3.2. *Let $\hat{x}^v_f$ be a solution of the linear system given by* (1.3.3) *satisfying the constraints in* (1.3.4). *Then $\bar{x}^v_f = \sum_{T \in \mathscr{T}_\Sigma} \hat{x}^{T(v)}_{T(f)}$ is also a solution of* (1.3.3) *satisfying* (1.3.9) *and* (1.3.4).

Proof. Let $T \in \mathscr{T}_\Sigma$. Since $\mathscr{T}_\Sigma$ is a group we have $\mathscr{T}_\Sigma = \{RT \mid R \in \mathscr{T}_\Sigma\}$. Hence

$$\bar{x}^{T(v)}_{T(f)} = \sum_{R \in \mathscr{T}_\Sigma} \hat{x}^{RT(v)}_{RT(f)} = \sum_{R \in \mathscr{T}_\Sigma} \hat{x}^{R(v)}_{R(f)} = \bar{x}^v_f. \tag{1.3.10}$$

Thus $\bar{x}^v_f$ satisfies (1.3.9).

We prove now that $\bar{x}_f^v$ is a solution of the linear system given by (1.3.3). For $e = ab \in \mathscr{E}_\Sigma$ we have $T(E_a) = E_{T(a)}$ because of the symmetric property of Voronoi diagrams (1.3.8), and hence of its dual. Therefore

$$\sum_{\substack{h \in E_a \\ h \multimap (\mathbb{S}_a \cap e)}} \hat{x}_{T(h)}^{T(a)} = \sum_{\substack{k \in E_{T(a)} \\ k \multimap (\mathbb{S}_{T(a)} \cap T(e))}} \hat{x}_k^{T(a)}. \tag{1.3.11}$$

Similarly

$$\sum_{\substack{g \in E_b \\ g \multimap (\mathbb{S}_b \cap e)}} \hat{x}_{T(g)}^{T(b)} = \sum_{\substack{l \in E_{T(b)} \\ l \multimap (\mathbb{S}_{T(b)} \cap T(e))}} \hat{x}_l^{T(b)}. \tag{1.3.12}$$

T is a skeleton symmetry, thus $T(e) \in \mathscr{E}_\Sigma$ is the edge connecting $T(a)$ and $T(b)$. Since $\hat{x}_f^v$ is a solution of the system given by (1.3.3), taking the equation for the edge $T(e)$ in (1.3.3), it follows that

$$\sum_{\substack{k \in E_{T(a)} \\ k \multimap (\mathbb{S}_{T(a)} \cap T(e))}} \hat{x}_k^{T(a)} = \sum_{\substack{l \in E_{T(b)} \\ l \multimap (\mathbb{S}_{T(b)} \cap T(e))}} \hat{x}_l^{T(b)}. \tag{1.3.13}$$

From (1.3.11), (1.3.12) and (1.3.13) we get

$$\sum_{\substack{h \in E_a \\ h \multimap (\mathbb{S}_a \cap e)}} \hat{x}_{T(h)}^{T(a)} = \sum_{\substack{g \in E_b \\ g \multimap (\mathbb{S}_b \cap e)}} \hat{x}_{T(g)}^{T(b)} \quad \forall e \in \mathscr{E}_\Sigma. \tag{1.3.14}$$

Summing the latter equation over $T \in \mathscr{T}_\Sigma$ proves that $\bar{x}_f^v$ is a solution of (1.3.3). □

Theorem 2. *For any skeleton Σ with a group of symmetries $\mathscr{T}_\Sigma$, the linear system given by* (1.3.3) *has a solution with the entries x_f^v $(v \in \mathscr{V}_\Sigma, f \in E_v)$ satisfying the constraints given in* (1.3.4) *and* (1.3.9)*. Therefore there exists a symmetric scaffold for Σ.*

Proof. By Theorem 1 we have that there is a solution $\hat{x}_f^v$ of (1.3.3) satisfying (1.3.4). Lemma 1.3.2 then gives a solution $\bar{x}_f^v$ to (1.3.3) satisfying (1.3.9). Since $\bar{x}_f^v \geq \hat{x}_f^v$, we have that (1.3.4) is trivially satisfied by $\bar{x}_f^v$. □

An example of a skeleton with a rotation symmetry is shown in Figure 1.6. If one is not interested in a solution that respects all the symmetries of the skeleton, one can restrict the set $\mathscr{T}_\Sigma$ to be the group generated by a subset of the skeleton symmetries.

1.3.4 Regular symmetric scaffold

We call a scaffold *regular* if it has the same number of quads around each line segment. This is not an automatic property of *standard* scaffolds, as can be seen in figure 1.6b and 1.7.

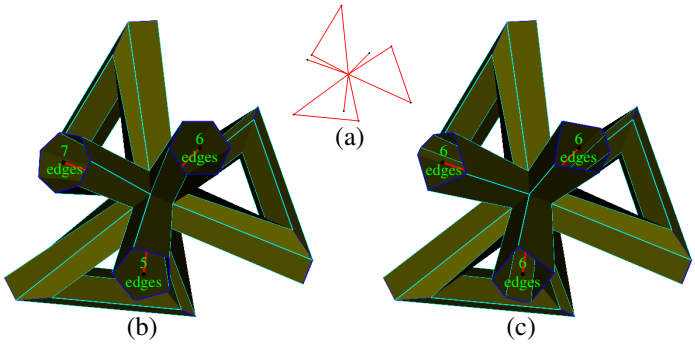


Figure 1.6: A three-fold rotation symmetry (C_3). (a) Skeleton. (b) Standard scaffold. (c) Symmetric scaffold.

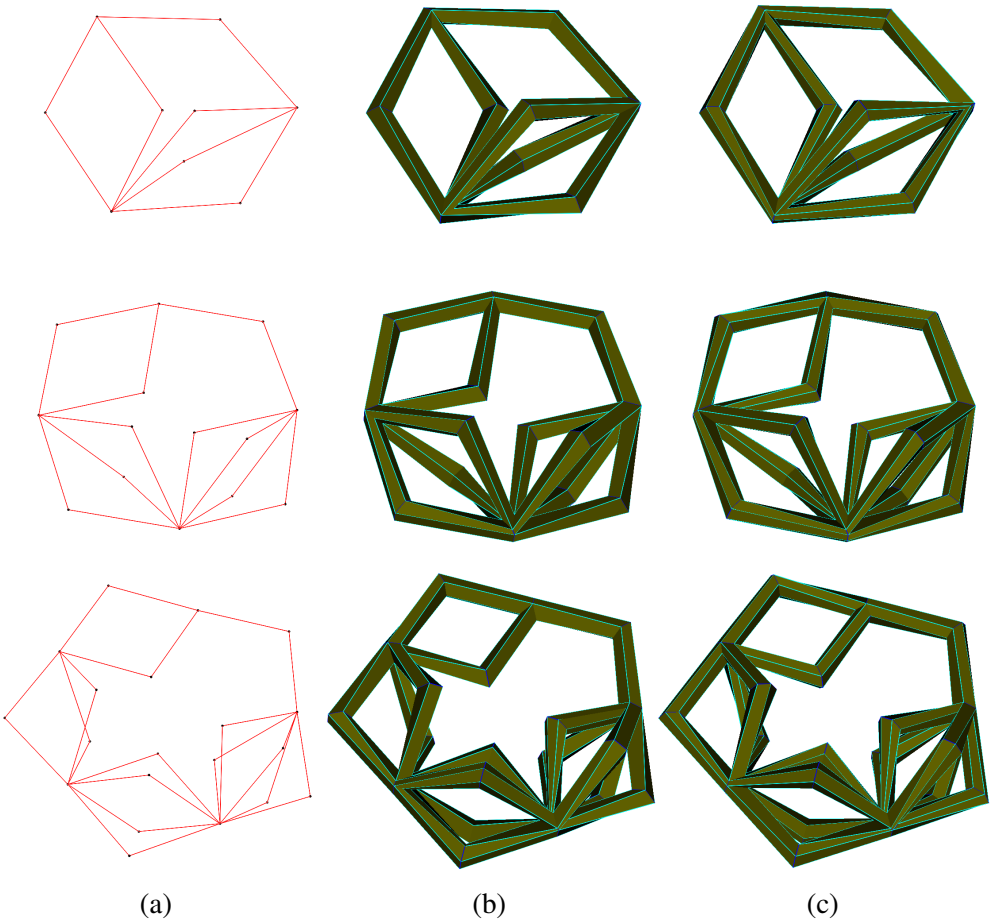


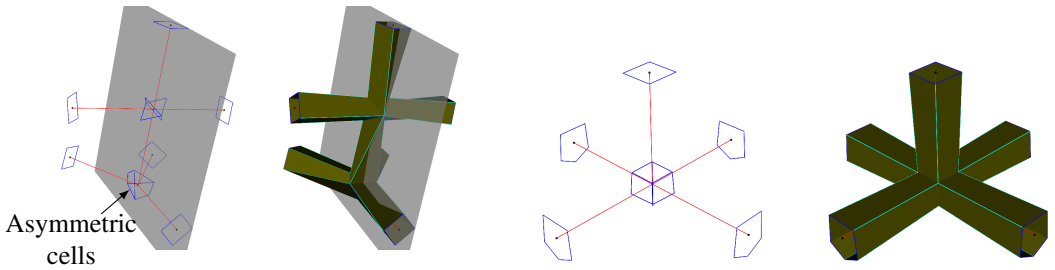
Figure 1.7: Complex closed skeletons (a), standard scaffolds (b), and regular scaffolds (b)

To get regular scaffolds we need compatible cells that have the same number of points, this means that (1.3.3) must be replaced by

$$\sum_{\substack{f\in E_v \\ f\multimap(\mathbb{S}_v\cap e)}} x_f^v = \lambda \quad \forall v\in\mathscr{V}_\Sigma, e\multimap v, e\in\mathscr{E}_\Sigma \tag{1.3.15}$$

with λ an additional free variable. Notice that λ is independent of $v\in\mathscr{V}_\Sigma$ and so (1.3.15) implies (1.3.3). Solutions to the linear system (1.3.15) satisfying (1.3.4) are called *regular* solutions. Regularity ensures that all segments have a similar cross profile: a polygon with λ sides. If there is a regular solution we say that (1.3.15) is feasible, which implies the existence of a regular scaffold.

It is possible to get a scaffold that is at the same time regular and symmetric. We just need to ensure, besides (1.3.4), the extra constraints in (1.3.9). A regular scaffold need not be symmetric (Figure 1.8a). Conversely, a symmetric scaffold is not necessarily regular (Figure 1.8b).



(a) (b)

Figure 1.8: Symmetry and regularity are independent properties. (a) Regular asymmetric scaffold: the highlighted cell is asymmetric with respect to the reflection symmetry. (b) Symmetric irregular scaffold: one edge has a quadrangular cross profile while the rest are pentagonal.

The existence of regular symmetric scaffolds is established in the following theorem. It is sufficient to prove the feasibility of *regular symmetric* scaffolds to automatically get the feasibility of *regular* scaffolds. Indeed a regular scaffold can be regarded as a regular symmetric scaffold with $\mathscr{T}_\Sigma$ the trivial group consisting of only the identity symmetry.

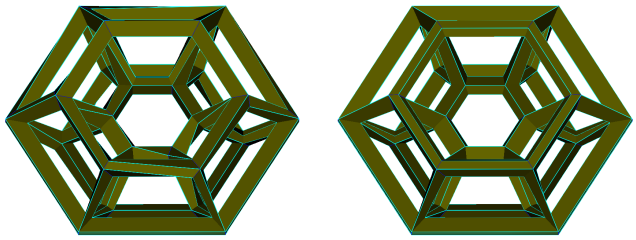
Theorem 3. *For any skeleton Σ admitting a group of symmetries $\mathscr{T}_\Sigma$, the linear system given by* (1.3.15) *has a solution with the entries x_f^v $(v\in\mathscr{V}_\Sigma, f\in E_v)$ satisfying the constraints given in* (1.3.4) *and* (1.3.9)*. Therefore there exists a regular symmetric scaffold for Σ.*

Proof. As done in the proof of Theorem 1, for every $v\in\mathscr{V}_\Sigma$ Lemma 1.3.1 gives a *local solution* $(\tilde{x}_f^v,\tilde{\lambda}_v)$ satisfying (1.3.1). Then $\hat{x}_f^v = s\frac{\hat{\lambda}}{\tilde{\lambda}_v}\tilde{x}_f^v$ with $\hat{\lambda}=\prod_{u\in\mathscr{V}_\Sigma}\tilde{\lambda}_u$ and $s=\max_i s_i$, is a local solution such that all the cells have $s\hat{\lambda}$ points, thus $\hat{x}_f^v$ is a global solution

of (1.3.15) satisfying (1.3.4). Applying Lemma 1.3.2 to this solution we get a symmetric solution $\bar{x}^v_f$ with the same number of points on each cell. Indeed

$$|\bar{C}^v_e| = \sum_{\substack{f\in E_v \\ f\multimap(\mathbb{S}_v\cap e)}} \bar{x}^v_f = \sum_{\substack{f\in E_v \\ f\multimap(\mathbb{S}_v\cap e)}} \sum_{T\in\mathscr{T}_\Sigma} \hat{x}^{T(v)}_{T(f)} = \sum_{T\in\mathscr{T}_\Sigma} \sum_{\substack{g\in E_{T(v)} \\ g\multimap(\mathbb{S}_{T(v)}\cap T(e))}} \hat{x}^{T(v)}_g = \sum_{T\in\mathscr{T}_\Sigma} |\hat{C}^{T(v)}_{T(e)}| = s\hat{\lambda}\,|\mathscr{T}_\Sigma|. \quad \square$$

Symmetry is an important property for the scaffolds as can be appreciated in Figure 1.9 computed for one of the examples in [Karčiauskas and Peters, 2016].



(a) (b)

Figure 1.9: With symmetric requirements a scaffold can be greatly improved. (a) The computation of the scaffold in Figure 1.1a without symmetric requirements. (b) The "correct" symmetric scaffold respecting all the symmetries of the skeleton (same as Figure 1.1a).

1.4 Optimal scaffolds

We have established that for any skeleton Σ admitting a group of symmetries $\mathscr{T}_\Sigma$, there exist a standard scaffold, a symmetric scaffold, a regular scaffold, and a regular symmetric scaffold. In this section we address the computation of optimal scaffolds. We model each scaffolding problem as an Integer Linear Program (IP) [Chen et al., 2009] that looks for a solution with a minimal number of quads. For this we need first to define an objective function, then the constraints of the IPs. The existence of a minimal solution is guaranteed by the existence theorems on Section 1.3, thus such minimal solutions can be computed with an IP solver.

1.4.1 Objective function

We want to express the total number of quads in a scaffold in terms of the subdivision of the arcs. For this we proceed as follows. Let $\mathscr{Q}$ be the set of all quads of a scaffold, and P the number of pairs $\langle p, Q\rangle$ such that $p \in Q \in \mathscr{Q}$. Then we have that

$$|\mathscr{Q}| = \tfrac{1}{4}P. \tag{1.4.1}$$

Indeed, the last equation is a consequence of the following observation: every fixed quad Q has 4 points and there are precisely 4 pairs $\langle q, Q\rangle$ with $p \in Q$.

We can count P in another way: by fixing first a point and then counting the pairs containing it. For a point p in the cell of a dangling node, there are 2 quads containing p. If p is a point in the cell of an articulation, there are 4 quads containing p. If p is a point in an arc on the boundary of a Voronoi region of a joint (not an articulation), we get that p is in 4 quads if p is not an extremity of the arc, or p is in $2d(p)$ quads if it is an extremity of an arc. Here $d(p)$ denotes the number of Voronoi regions that have p in their common boundary.

The latter paragraph can be summarized as

$$P = \sum_{\substack{v \in L_\Sigma \\ f \in E_v}} 2x_f^v + \sum_{\substack{v \in M_\Sigma \\ f \in E_v}} 4x_f^v + \sum_{\substack{v \in N_\Sigma \\ f \in E_v}} 4(x_f^v - 1) + \Theta_\Sigma, \tag{1.4.2}$$

where Θ_Σ is the number of pairs containing points that are extremities of the arcs in the Voronoi diagrams. Notice that Θ_Σ is a constant for a fixed skeleton Σ, thus (1.4.2) can be written as

$$P = \sum_{\substack{v \in L_\Sigma \\ f \in E_v}} 2x_f^v + \sum_{\substack{v \in M_\Sigma \\ f \in E_v}} 4x_f^v + \sum_{\substack{v \in N_\Sigma \\ f \in E_v}} 4x_f^v + \Phi_\Sigma, \tag{1.4.3}$$

where $\Phi_\Sigma = \Theta_\Sigma - \sum_{v \in N_\Sigma} 4|E_v|$ is a constant for a fixed skeleton Σ.

From equations (1.4.1) and (1.4.3) we conclude that

$$\sum_{v \in (\mathcal{V}_\Sigma - L_\Sigma)} \sum_{f \in E_v} 2x_f^v + \sum_{v \in L_\Sigma} \sum_{f \in E_v} x_f^v \tag{1.4.4}$$

is an objective function that minimizes the total number of quads in a scaffold.

1.4.2 Integer Linear Programming models

We can now state the Integer Linear Programs (IPs) that compute subdivision for the arcs such that the scaffold have a minimal number of quads. For all the IPs the optimality criterion is to minimize the objective function (1.4.4). In Table 1.1 we show each model.

Theorems 1, 2 and 3 prove the feasibility of the standard, symmetric, and regular symmetric models respectively. As discussed in Section 1.3.4, the feasibility of the regular IP model is a consequence of the feasibility of the regular symmetric model.

A final observation is that once we can guarantee feasibility, and since all the variables in (1.4.4) are positive integers, then an optimal minimal solution always exists. The optimal solution can be computed with a Mixed-Integer Linear Program Solver. In particular with the branch-and-cut [Chen et al., 2009, Chapter 12] implementation of GLPK [Makhorin, 2016].

Scaffold	Constraints	Minimize function
Standard	(1.3.3), (1.3.4)	(1.4.4)
Regular	(1.3.15), (1.3.4)	(1.4.4)
Symmetric	(1.3.3), (1.3.4), (1.3.9)	(1.4.4)
Regular Symmetric	(1.3.15), (1.3.4), (1.3.9)	(1.4.4)

Table 1.1: Integer Linear Program models.

1.5 Algorithms

In this section we provide practical algorithms for the implementation of our method. Although our main contribution is in the theoretical foundation and the proofs of the existence of scaffolds, the algorithms in this section show that our method can be readily implemented to compute scaffolds for any skeleton.

The general steps for an implementation are described in Algorithm 1. Sub-algorithms 2, 3 and 4 deal with the details of the construction of compatible cells from the output of the standard IP and the definition of the bijections between cells (linking process).

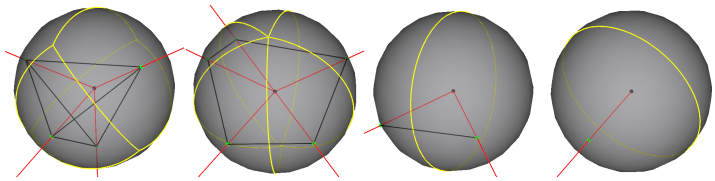
Input: The set of nodes $\mathscr{V}_\Sigma$ and edges $\mathscr{E}_\Sigma$ representing the skeleton.
Output: The quads that represent a scaffold.

1 **foreach** *node* $v \in \mathscr{V}_\Sigma$ **do**
2 $\quad \mathscr{A}_v \leftarrow \{e \cap \mathbb{S}_v : e \in \mathscr{E}_\Sigma, e \multimap v\}$;
3 $\quad \mathscr{H}_v \leftarrow convexHull(\mathscr{A}_v)$;
4 $\quad E_v \leftarrow edgesOf(\mathscr{H}_v)$;
5 Define and solve the Linear Program for the subdivision of arcs; `/* Alg. 2 */`
6 Construct compatible cells; `/* Alg. 3 */`
7 Define bijections of linked cells; `/* Alg. 4 */`
8 **foreach** *edge* $e = ab \in \mathscr{E}_\Sigma$ **do**
9 $\quad$ Let $C_e^a = \langle p_0, p_2, \ldots, p_n \rangle$;
10 $\quad$ **for** $i = 0$ **to** n **do**
11 $\quad\quad$ Output quad $\langle p_i, \phi_e(p_i), \phi_e(p_{i+1}), p_{i+1} \rangle$ `/* i+1 is taken mod n */`

Algorithm 1: General algorithm for constructing a scaffold.

Algorithm 1 follows the general three steps we introduced in Section 1.1. In Step 1 we use spherical Voronoi diagrams for the partition of the sphere at the joints (lines 1–4). We mostly work with their duals, Delaunay triangulations, which are equivalent to the the convex hulls [Brown, 1979, Grima and Márquez, 2001]. The discretization of the regions, Step 2, is divided into sub-algorithms.

The convex hull computed in Algorithm 1 (line 3) is not always 3-dimensional. It may be a 2-dimensional convex hull (a polygon), a 1-dimensional convex hull (one edge), or a 0-dimensional one (a point). Those cases correspond respectively to: more than three points in general position, more than two coplanar points, two points, and only one point (Figure 1.10). 1-dimensional convex hulls occur around articulations, while 0-dimensional ones are associated to dangling nodes. In our implementation the convex hull is computed by means of the QHull library [Barber et al., 1996].



(a) (b) (c) (d)

Figure 1.10: Cases for the convex hull of $\mathscr{A}_v$. Edges of E_v are shown in black. (a) 3-dimensional: more than three points in general position. (b) 2-dimensional: more than two coplanar points. (c) 1-dimensional: two points. (d) 0-dimensional: one point.

Algorithm 2 deals with the standard IP whose optimal solutions give compatible cells. It sets up and solves the IP using the convex hull representation of the regions. Simple modifications can be done for the other three variants of the scaffold. We follow (1.3.5). Thus we chose to have at least 4 points on each cell to guarantee at least quadrangular cross sections as in [Bærentzen et al., 2012, Ji et al., 2010, Panotopoulou et al., 2018, Usai et al., 2015, Yao et al., 2009]. We set the threshold for *long* arcs at $\frac{5\pi}{6}$ (i.e $\delta = \frac{\pi}{6}$), above which at least two subdivisions must be done. In our implementation we solve the IP with the branch-and-cut [Chen et al., 2009, Chapter 12] implementation in GLPK [Makhorin, 2016].

Algorithm 3 computes the cells from the subdivision numbers and the convex hull representation of the Voronoi diagram, and such it must handle the different cases of the convex hulls (see Figure 1.10).

The heuristic for the linking process (Step 3) is described in Algorithm 4, from which the resulting quads are defined. It defines the bijections (links) by minimizing the total length of the links (same heuristic used in [Bærentzen et al., 2012]).

To better reflect the geometry of the model, and depending on the application at hand, the position of points in the discretization of each arc can be modified by a Laplacian smoothing [Botsch et al., 2010] or another heuristic. A global optimization could be further applied to take into account the twisting of the quads around a chain of articulation nodes (Figure 1.11). The challenge is in devising such a "better" positioning of points that also respects the symmetries of the skeleton. An intrinsic solution using our method is to subdivide each arc twice (or more) as shown in Figure 1.11.

Input: The set of nodes $\mathscr{V}_\Sigma$ and edges $\mathscr{E}_\Sigma$ representing the skeleton, along with E_v for each $v \in \mathscr{V}_\Sigma$.

Output: The values x_f^v representing the number of subdivisions for each arc that gives compatible cells.

Initialize the linear program IP;
foreach *node* $v \in \mathscr{V}_\Sigma$, *and edge* $f \in E_v$ **do**
    Add integer variable x_f^v to IP;
    if *the arc associated to* x_f^v *has length* $< \frac{5\pi}{6}\varepsilon_v$ **then**
        Add restriction $x_f^v \geq 1$ to IP;
    else
        Add restriction $x_f^v \geq 2$ to IP;
foreach *pair* (v,e) *with* $v \in \mathscr{V}_\Sigma$, $e \in \mathscr{E}_\Sigma$ *and* $e \multimap v$ **do**
    Add following restriction to IP $\sum_{\substack{f \in E_v \\ f \multimap (e \cap \mathbb{S}_v)}} x_f^v \geq 4$;
foreach *edge* $e = ab \in \mathscr{E}_\Sigma$ **do**
    Add following restriction to IP $\sum_{\substack{g \in E_a \\ g \multimap (e \cap \mathbb{S}_v)}} x_g^a = \sum_{\substack{h \in E_b \\ h \multimap (e \cap \mathbb{S}_v)}} x_h^b$;
Define objective function $\sum_{v \in (\mathscr{V}_\Sigma - L_\Sigma)} \sum_{f \in E_v} 2x_f^v + \sum_{v \in L_\Sigma} \sum_{f \in E_v} x_f^v$ for IP;
Solve IP by minimizing the objective function.

Algorithm 2: Compute subdivisions for compatible cells.

Input: Nodes $\mathscr{V}_\Sigma$ and edges $\mathscr{E}_\Sigma$ of the skeleton, E_v, $\mathscr{H}_v$ and the subdivision numbers x_f^v.
Output: A list $\mathscr{C}$ of compatible cells for the scaffold.

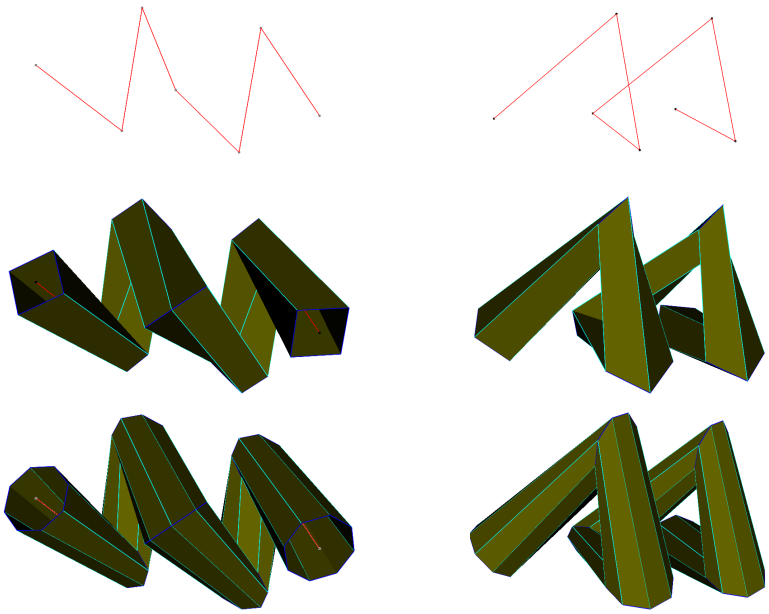
$\mathscr{C} \leftarrow emptyList()$;
foreach $v \in \mathscr{V}_\Sigma$ **do**
  if $\mathscr{H}_v$ *is 3-dimensional* **then**
    /* at least 4 not coplanar nodes */
    foreach *node* n_e *in* $\mathscr{H}_v$ **do** /* $n_e = e \cap \mathbb{S}_v$ for $e \multimap v$ */
      $C_e^v \leftarrow emptyList()$;
      foreach $f \multimap (e \cap \mathbb{S}_v)$ **do**
        Let F_1, F_2 be the two faces in $\mathscr{H}_v$ that have common boundary f;
        Compute the outward pointing unit normals N_1, N_2 of F_1, F_2 respectively;
        Compute $x_f^v + 1$ points in the arc from $v + N_1$ to $v + N_2$ going perpendicularly to f on $\mathbb{S}_v$;
        Add the points to C_e^v avoiding repetitions;
      Add C_e^v to $\mathscr{C}$;
  if $\mathscr{H}_v$ *is 2-dimensional* **then**
    /* at least 3 nodes, all coplanar */
    Let N_1, N_2 be the two unit normals of the plane supporting $\mathscr{H}_v$;
    foreach *node* n_e *in* $\mathscr{H}_v$ **do** /* $n_e = e \cap \mathbb{S}_v$ for $e \multimap v$ */
      $C_e^v \leftarrow emptyList()$;
      Let f, g be the two edges incidents to n_e;
      Compute $x_f^v + 1$ points in the arc from $v + N_1$ to $v + N_2$ going perpendicularly to f on $\mathbb{S}_v$;
      Compute $x_g^v + 1$ points in the arc from $v + N_1$ to $v + N_2$ going perpendicularly to g on $\mathbb{S}_v$;
      Add the points to C_e^v avoiding repetitions;
      Add C_e^v to $\mathscr{C}$;
  if $\mathscr{H}_v$ *is 1-dimensional* **then**
    /* only one edge and 2 nodes */
    Let f be the unique edge in E_v;
    Compute x_f^v points in the circle with center v and perpendicular to f on $\mathbb{S}_v$;
    Add the points to C_e^v;
    Add C_e^v to $\mathscr{C}$;
  if $\mathscr{H}_v$ *is 0-dimensional* **then**
    /* only one node */
    Compute $x_{v_v}^v$ points in the great circle with center v and perpendicular to e on $\mathbb{S}_v$;
    Add the points to C_e^v;
    Add C_e^v to $\mathscr{C}$;
return $\mathscr{C}$

Algorithm 3: Construct compatible cells.

Input: The set of compatible cells $\mathscr{C} = \{C_e^v \mid v \in \mathscr{V}_\Sigma, e \in \mathscr{E}_\Sigma\}$.
Output: The bijections ϕ_e between linked cells.

foreach *edge* $e = ab \in \mathscr{E}_\Sigma$ **do**
  Order points in the cells C_e^a and C_e^b according to the angle around the edge e;
  Let $C_e^a = \langle p_0, p_2, \ldots, p_n \rangle$ and $C_e^a = \langle q_0, q_2, \ldots, q_n \rangle$;
  $D_{min} \leftarrow \sum_{i=0}^{n} \|p_i - q_i\|$; $k \leftarrow 0$;
  for $j = 1$ **to** $n-1$ **do**
    $D \leftarrow \sum_{i=0}^{n} \|p_i - q_{i+j}\|$;
    if $D < D_{min}$ **then**
      $D_{min} \leftarrow D$; $k \leftarrow j$;
  Define the bijection ϕ_e as $p_i \mapsto q_{i+k}$;
  /* $i+k$ is taken mod n */

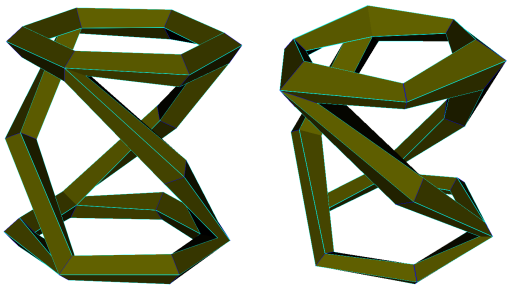
Algorithm 4: Construct bijections between linked cells.



(a) (b)

Figure 1.11: In a chain of articulations the scaffold might have a twisting behavior: (a) and (b) – different views of the same model; top row – the skeleton; middle row – the standard scaffold; and bottom row – the scaffold with extra subdivisions. Note how in the bottom row the twisting behavior of the quads is diminished.

An example of scaffold with different radii for the spheres at the joints is shown in Figure 1.12. Note that the skeleton is symmetric as well as the scaffold, hence the values of the radii are also symmetric.



(a) (b)

Figure 1.12: A scaffold with different radii for the spheres at the joints: (a) the original scaffold, (b) the scaffold with varying radii.

Symmetric and regular scaffolds Regular scaffolds can be obtained with the algorithms described above, a simple variation of Algorithm 2 is needed in order to account for the restrictions given in (1.3.15). Similarly, for symmetric scaffolds we need to add the extra restrictions in (1.3.9), taking special care with the definition of links (Algorithm 4). Dangling nodes or articulations fixed by the skeleton symmetries might present issues if the linking is naively done. In our implementation we modify Algorithm 4 such that the links are propagated from one edge to its symmetric ones. The cells of dangling nodes are computed by projecting the linked cells onto the plane perpendicular to the incident edge (of the dangling nodes). This guarantees that the links respect the symmetries.

Linear system simplification The IP in Algorithm 2 can be simplified by removing variables and equations relative to dangling nodes. The equations in a chain of articulations can be merged into one equation relating the extremal cells of the chain, removing the variables relative to intermediate articulations. In Figure 1.13 we illustrate the simplification cases.

The general complexity of our method is bounded by the IP solver algorithm, which theoretically is NP-complete [Karp, 2009]. In practice, as usual with Linear Programming, the solver behaves well. The convex hull implementation has average complexity $n \log n$, with n the number of skeleton edges. In Table 1.2 we show a summary of the running time of our method for some examples. It illustrates the general relation between the total time and the time spent in solving the IP. The reported times are averages of 200 runs.

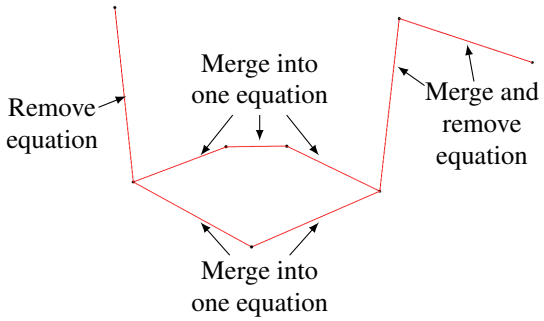


Figure 1.13: Simplification of the equations in the IP.

	Total	**IP**	**Other**		**Total**	**IP**	**Other**
Fig. 1.9a	133	14	119	Fig. 1.9b	142	19	123
Fig. 1.6b	55	12	43	Fig. 1.6c	55	12	43
Fig. 1.7b-top	16	4	12	Fig. 1.7c-top	24	4	20
Fig. 1.7b-mid	29	6	23	Fig. 1.7c-mid	37	7	30
Fig. 1.7b-bot	44	7	37	Fig. 1.7c-bot	51	6	45

Table 1.2: Running time (in milliseconds) of our implementation. Columns: **Total** – total running time, **IP** – time for finding the IP solution, **Other** – rest of the time.

Our Python implementation is not optimal but versatile. It generates the IP problem file and call GLPK [Makhorin, 2016] (in particular the utility `glpsol`) to solve the IP. QHull [Barber et al., 1996] is also called as an external utility and has a negligible running time. Better timings should be expected from a production-ready C++ implementation with direct linking to GLPK and QHull libraries.

1.6 Further simplifications of the scaffold

As we discussed in Section 1.3, the minimal number of points for each cell can be set to three. This decreases the number of quads in the final scaffold and might generate triangular cross-sections along some skeleton edges. In Figure 1.14 we recompute the example in Figure 1.1a with the new lower bounds, this decreases the number of quads in the standard scaffold from 192 to 144.

Table 1.3 summarizes the number of quads and vertices per valency of some of the scaffolds in this paper, all of which are optimally computed with our implementation. It shows that the symmetric scaffolds on Figure 1.4 and 1.6 are also optimal standard scaffolds.

Another simplification arises by noticing that any partition of the sphere with an

	v_3	v_4	v_6	t_{quads}		v_3	v_4	v_6	t_{quads}
Fig. 1.14	*	*	96	144	Fig. 1.6b	18	33	14	63
Fig. 1.1a	*	48	96	192	Fig. 1.6c	18	33	14	63
Fig. 1.1b	*	120	*	120	Fig. 1.16b	48	66	12	108
Fig. 1.1c	*	72	*	72	Fig. 1.16c	96	150	12	216
Fig. 1.1d	*	72	*	72	Fig. 1.16d	192	318	12	432
Fig. 1.4a-b	24	8	4	24	Fig. 1.16e	384	654	12	864

Table 1.3: Summary of some scaffolds in this chapter. Columns: $\mathbf{v_k}$ – number of vertices with valency $\mathbf{k}$, $\mathbf{t_{quads}}$ – total number of quads. The entries $*$ mean zero.

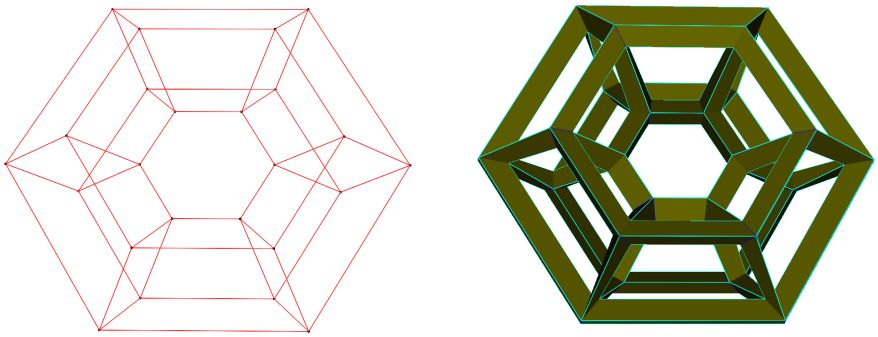
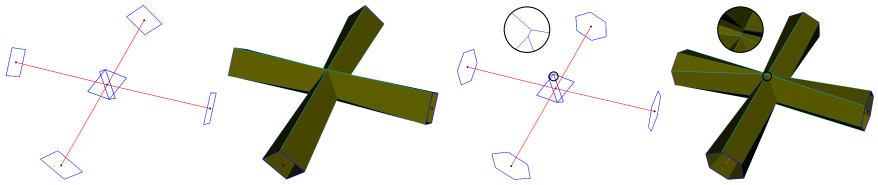


Figure 1.14: Example of a scaffold constructed with 3 as minimal number of points on each cell. Compare with Figure 1.1a.

inscribable dual gives feasible solutions for the IPs. Thus the computation of Voronoi regions can be modified by changing the minimal angle for which two triangles sharing an edge in the convex hull are considered to be coplanar. In our experience, this helps to reduce the complexity of the cells around joints and in general reduces (even more) the number of quads needed to construct a scaffold. As an example, in Figure 1.15 four close-to-coplanar points can be considered coplanar, yielding compatible cells with less points (hence less quads). On the other hand, due to round-off errors coplanar points may appear as not coplanar. This issue provokes that very small arcs appear on the boundary of Voronoi regions, consequently some quads may look like triangles at a large scale (Figure 1.15b).



(a) (b)

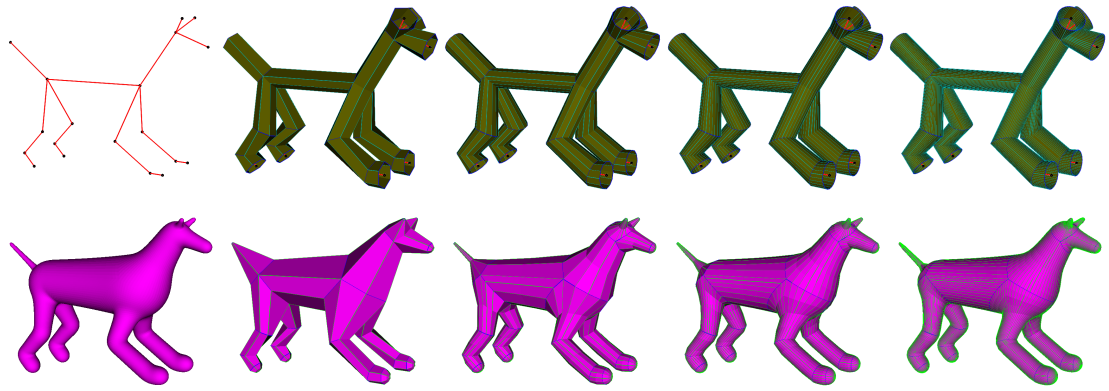
Figure 1.15: Close-to-coplanar points can be treated as coplanar as a way to improve the scaffold. In the picture the four intersection points in the joint are not coplanar. In (a) they are treated as coplanar points, in (b) the exact Voronoi regions (and cells) are computed.

1.7 Application to polygonization

In this paper we focus on the feasibility of the scaffold construction, including the symmetric and regular variants. The scaffold is in most applications an *intermediate* step [Bærentzen et al., 2012, Bærentzen et al., 2014, Blidia et al., 2017, Ji et al., 2010, Karčiauskas and Peters, 2016, Srinivasan et al., 2005, Usai et al., 2015, Yao et al., 2009]. Although we present theoretical results and practical algorithms we also want to showcase an example application.

Polygonization of implicit surfaces is usually done with Marching Cubes algorithm [Lorensen and Cline, 1987] or one of its variants [Wenger, 2013, Chapter 2]. For computer graphics applications quad-dominant meshes are more desirable and this requires a re-meshing of the standard triangular meshes obtained with the Marching Cubes algorithms [Botsch et al., 2010, Section 6.6]. For articulated models, it is a good improvement if the quad mesh follows the structure of the skeleton. A scaffolding technique can be used to bypass the re-meshing process and get directly a quad-dominant mesh that follows the skeleton.

Figure 1.16 shows a mesh computed with our scaffolding technique. The implicit surface (Figure 1.16a) is a convolution surface [Zanni, 2013] for which a scaffold is



(a) The model. (b) Standard ($\kappa = 1$). (c) $\kappa = 2$. (d) $\kappa = 4$. (e) $\kappa = 8$.

Figure 1.16: The scaffold method as a basis for a quad dominant mesh of an implicit surface. (a) Top: the skeleton, bottom: an implicit convolution surface around the skeleton. The rest of the columns (b-e) represent the standard scaffold solution (of subdivision of arcs for compatible cells) multiplied by the constant κ. In (b-e) κ also defines the number of quads along each line segment.

computed with a higher number of subdivisions for each arc: multiplying the standard solution by an integer $\kappa > 0$ that also defines the number of quads along a segment. To obtain the surface mesh we project the scaffold quads onto the surface. This is done by ray-shooting: compute the intersection with the surface of rays emanating from the skeleton in the direction given by quad vertices. A similar projection was done in [Angelidis and Cani, 2002], the main difference with our approach is that the mesh we project (scaffold) is computed for the whole skeleton while in [Angelidis and Cani, 2002] a fixed set of meshes is defined for each line segment. At the tips of dangling nodes, the quads are computed by creating a polar-annular region with a singular point (as done in [Bærentzen et al., 2012]). The polar-annular meshes transform all valency 3 vertices on the boundary of the scaffold mesh into valency 4 vertices at the cost of having an extra high valency vertex per dangling node.

1.8 Conclusions

We presented a method to construct a coarse quad-dominant mesh around a skeleton made of line segments that can serve as *intermediate* step in several applications. The mesh follows the structure of the skeleton and is computed in an optimal way, minimizing the total number of quads. Our method works for any skeleton even in the presence of cycles. We modeled the problem as an Integer Linear Program and proved its feasibility.

We presented variants of our method: we can generate a mesh with the same number of quads around each line segment, or a mesh with the same symmetries as the underlying skeleton. It is also possible to satisfy both conditions at the same time. The scaffold thus obtained can be used to polygonize implicit surfaces around the skeleton or as an intermediate step in other applications. Our method overcomes some limitations of, and add extra features to, previous work. The algorithmic description of each step gives a basis on which multiple implementations can be developed.

Chapter 2

Further topics on scaffolding

In this chapter we discuss further topics related to scaffolding: volumetric mesh generation based on hexahedra, and cell size control with spherical Laguerre diagrams. We discuss our early stage research with some examples illustrating applications, main issues, and challenges. The ideas in this chapter point to possible research lines for future work.

2.1 Hexahedral meshes

A scaffold can be used to generate a hexahedral (volumetric) mesh following an approach similar to [Livesu et al., 2016]. For each quad Q, let $e = ab$ be the edge in $\mathscr{E}_\Sigma$ along which Q is defined. Let $\langle p_0, p_1, p_2, p_3 \rangle$ be the (counterclockwise) vertices of Q, such that $\{p_0, p_1\} \subset C_e^a$, and $\{p_2, p_3\} \subset C_e^b$. Then one can define an hexahedron H_Q from Q as illustrated in Figure 2.1. The vertices of H_Q are given by $\langle a, b, p_0, p_1, p_2, p_3, q_a, q_b \rangle$ where q_a is the midpoint of the arc joining p_0 and p_1 (counterclockwise from the point of view of $e = ab$) and, respectively, q_b is the midpoint of the arc joining p_2 and p_3 (clockwise from the point of view of $e = ab$). The faces of H_Q are then $\{\langle a, b, p_2, p_1 \rangle, \langle b, a, p_0, p_3 \rangle, \langle a, p_1, q_a, p_0 \rangle, \langle b, p_3, q_b, p_2 \rangle, \langle p_1, p_2, q_b, q_a \rangle, \langle p_3, p_0, q_a, q_b \rangle\}$.

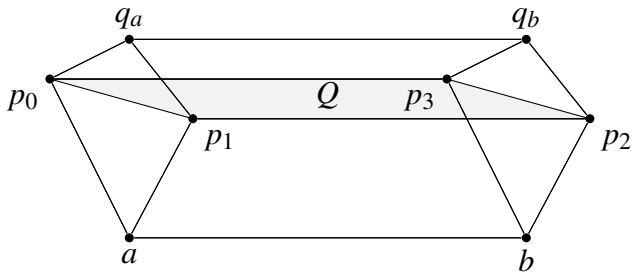
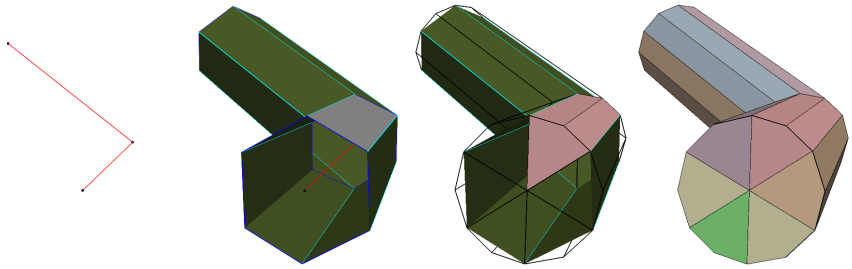


Figure 2.1: The hexahedron H_Q from the scaffold quad Q.

Repeating this process we can transform a scaffold into a hexahedral mesh as shown in Figure 2.2. The volumetric mesh can be subdivided further to get a smaller hexahedra by adding concentric layers from the supporting line segment ($e = ab$). In Figure 2.3 we illustrate a possible subdivision. Other subdivision patterns could be used (see, for example, [Livesu et al., 2017]).

The process described above (see Figure 2.1 and 2.2) always produces valid hexahedral meshes. This is mainly due to the fact that each quad patch in the base scaffold mesh corresponds to one, and only one, line segment on the skeleton. Hence a quad patch and its corresponding line segment uniquely defines a hexahedron as in Figure 2.1. On the other hand, since there are no spurious quads, the hexahedra cleanly partition the interior of the spheres at joints (see Figure 2.4e). This well behaved interior partition

is a consequence of the essential properties of spherical Voronoi diagrams: the region boundaries are composed of arcs of great circle.



(a) (b) (c) (d)

Figure 2.2: Hexahedral mesh from scaffold. Starting from a skeleton (a) we construct a scaffold (b). For each quad Q we construct a hexahedron. In (b) we highlighted a quad in gray that is converted into a hexahedron in (c). Repeating this process for every quad of the scaffold we get a volumetric mesh made of hexahedra (d).

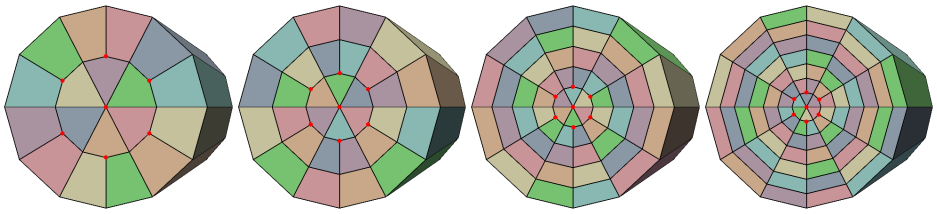
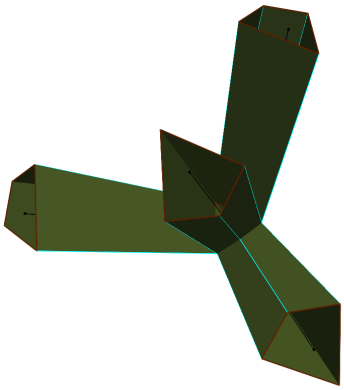
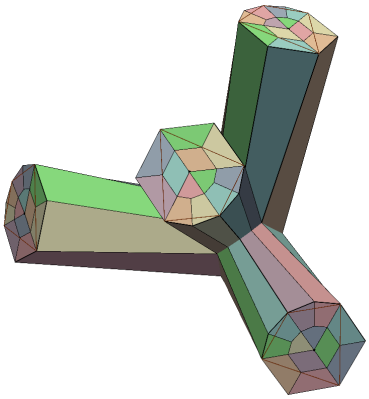


Figure 2.3: Stratified hexahedra subdivision. In red we highlight the singularities of the hexahedral mesh. Notice how the singularities cluster together close to the skeletons when the number of layers increases.

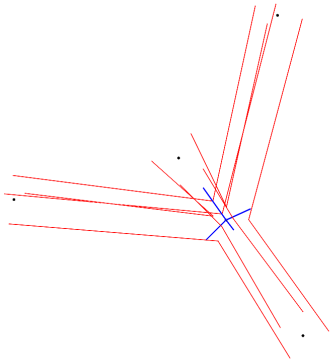
The *singular lines* in the hexahedral mesh are defined as the polylines made of mesh edges that do not have exactly four hexahedra around. In Figure 2.4 we show how the singular lines look like on the inside of the mesh. Singularities arise in three areas: the skeleton, the partition of the sphere at the joints, and clustered around the skeleton as a result of the hexahedral mesh subdivision scheme. A line segment skeleton is part of a singular line when the base scaffold of the hexahedral mesh does not have exactly four quads around the line segment skeleton (which is the most common case). On the other hand the singular lines given by the Voronoi partition of the sphere are predefined by the position of the Voronoi vertices, i.e. the extrema of the arcs in the boundary of Voronoi regions. They are given by the relative position of the incident line segments at the joint. Scaffold subdivisions may introduce a singular line by changing the number of quads around a line segment. The singularities defined by the partition at joints are unaffected by scaffold subdivisions.



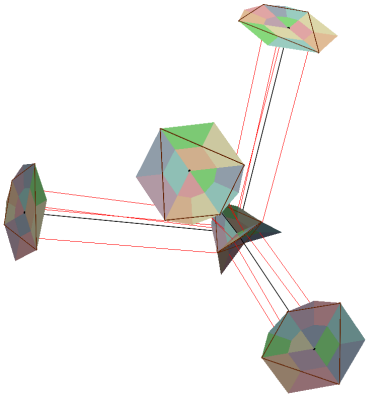
(a) Scaffold.



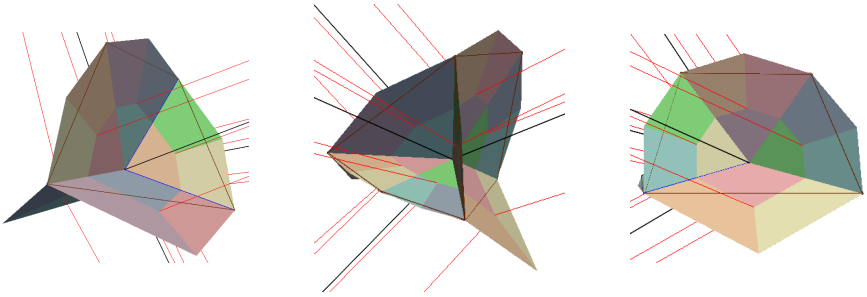
(b) Hexahedral mesh.



(c) Singular lines.



(d) Hexahedra borders.



(e) Zoomed view of the joint.

Figure 2.4: Hexahedra after a subdivision of two layers around a skeleton joint. From (a) and (b) one can appreciate how the scaffold is subdivided. In (c) and (d) we show the singular lines inside the hexahedral mesh: in red the singular lines given by the layering based on the scaffold, in blue the singular lines given by the Voronoi vertices on the partition of the sphere at the joint. Note that parts of the skeleton (in black) may also be a singular line when the scaffold has more than four patches around a skeleton piece (see Figure 2.2 and 2.3). In (e) one can appreciate the relation between the red singular lines and the patches of the base scaffold (brown lines).

Hexahedral meshes have several advantages that are out of the scope of this work. We want to illustrate nonetheless one application that greatly benefits from the underlying skeleton and its symmetric structure. In [Panetta et al., 2015, Panetta et al., 2017] a tetrahedral mesh is constructed for cubical pieces generated procedurally from a skeleton that satisfies the set of cube symmetries. In those papers the tetrahedral mesh is generated with a process that is unaware of the symmetries in the skeleton. We illustrate in Figure 2.5 some of the highly symmetric volumetric meshes generated with our method from examples in [Panetta et al., 2015, Panetta et al., 2017].

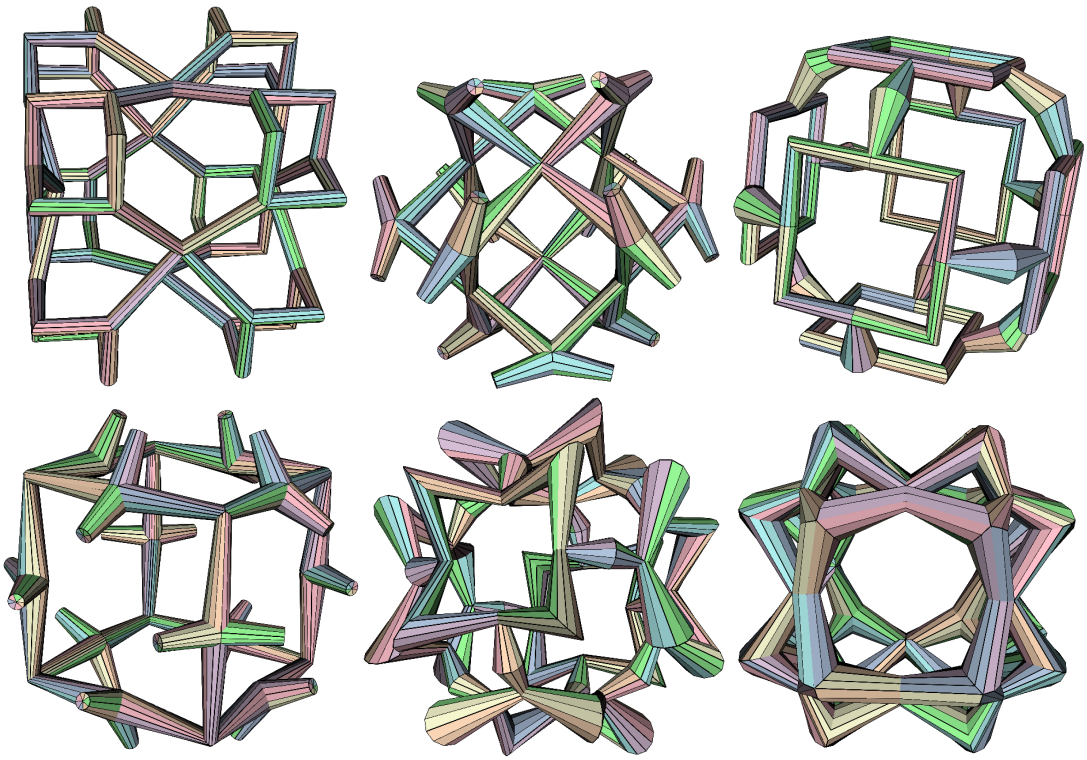


Figure 2.5: Hexahedral meshes for highly symmetric skeletons. The examples in the picture show meshes without subdivision. The symmetric structure of the final meshes can be appreciated.

2.1.1 Limitations

Although we can provide well defined hexahedral meshes with symmetry-respecting properties in an automatic way, our method has some limitations. First, the hexahedra so obtained may have sharp corners. That is the case for the edges supported on the line segments of the original skeleton. Sharp corners arise also from the configuration of the spherical Voronoi diagram partitions and the quads on the scaffold. Secondly,

the subdivision scheme described before is homogeneous for all parts of the skeleton. This means that the thickness of the layers varies with the thickness of the scaffolds, hence producing different sized hexahedra depending on the local thickness around the skeleton. These issues may be solved by proposing a more advance subdivision scheme for the original coarse hexahedral mesh. If an adaptive layering approach was to be followed, special care must be taken around joints at the interface between hexahedra generated from different line segments of the skeleton. Another issue is the presence of singularities given by the Voronoi vertices, over which we have no control.

2.2 Laguerre diagrams: controlling the cell size

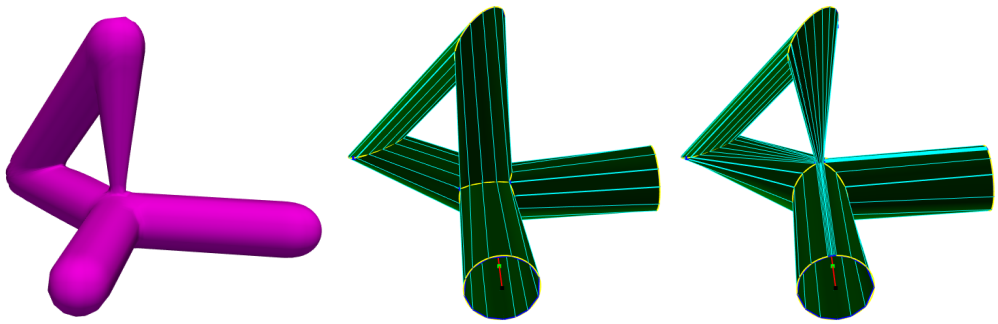
In Chapter 1 we discussed how to construct a scaffold for line segment skeletons. We showed that the radius of the sphere at a joint can be picked arbitrarily (as long as one respects the symmetries of the model) and still we get a valid scaffold. In Chapter 5 we improved the scaffold construction to handle general curves by exploiting circular splines and tangential polylines. In this section we are going to discuss another interesting extension that addresses the issue of different incident radii at joints. This is the case, for example, when modeling with convolution surfaces: the radius of the part being modeled around each incident line segment does not need to be equal for all incident pieces. In anisotropic convolution this may go to the extreme case where there are ellipsoids (hence three radii per incident segment) and different orientations. We do not treat here the ellipsoids case but we can do something for the case of different incident radii (without anisotropy). Figure 2.6a shows an example surface with a joint with different radii on the incident segments.

The solution we propose is to use spherical Laguerre diagrams instead of a spherical Voronoi diagrams. Generally speaking a Laguerre diagram is a variation of the Voronoi diagram that permits to *influence* the size of the cells without moving the sites. In Figure 2.6 we show how this may be used. When using spherical Laguerre diagrams the general method of scaffolding can be reused, with some caveats that we discuss later in this section. In terms of algorithms we present the modifications needed in order to compute a spherical Laguerre diagram using the dual convex hull. The method we follow was described and proved in [Sugihara, 2002] using clever and clear geometrical constructions.

2.2.1 Definition and algorithm variants

Before discussing further we recall first the definition of Laguerre diagram in the plane. Given a set of points $\{A_1, A_2, \ldots, A_n\} \subset \mathbb{R}^2$ and a set of weights $\{w_1, w_2, \ldots, w_n\} \subset \mathbb{R}$ we define the *Laguerre region* $\mathscr{L}_i$ ($i = 1, 2, \ldots, n$) as

$$\mathscr{L}_i = \{P \in \mathbb{R}^2 : d_L(P, A_i) \leq d_L(P, A_j)\ \forall j = 1, 2, \ldots, n\ j \neq i\} \tag{2.2.1}$$



(a) Surface (b) Voronoi (c) Laguerre

Figure 2.6: A spherical Laguerre diagram can be used to get a scaffold that better represents the radii of the incident segments. In the picture we show: (a) an example surface with different incident radii, (b) the (regular symmetric) scaffold computed with spherical Voronoi diagrams, and (c) the (regular symmetric) scaffold computed with spherical Laguerre diagrams.

where d_L is the *Laguerre distance*[1] defined by

$$d_L(P,A_i) = \|P - P_i\|^2 - w_i^2. \tag{2.2.2}$$

The collection of Laguerre regions (excluding their boundaries) partition the plane, and this partition is what we understands as *Laguerre diagram*. Note that if we take $w_i = 0$ for all i then we recover a partition that is the Voronoi diagram of $\{A_i\}_{i=1}^n$. Laguerre diagrams can be interpreted as the generalization of Voronoi diagrams for circles with center P_i and radius w_i.

The advantage of the Laguerre diagram is that we can change the shape and size of the regions $\{\mathscr{L}_i\}_{i=1}^n$ by picking appropriate values for $\{w_i\}_{i=1}^n$. Although we can now influence the size of regions there are some limitations when compared to Voronoi diagrams: (i) it may happen that some regions are empty, i.e. they "disappear" from the diagram; (ii) it is not always true that $A_i \subset \mathscr{L}_i$, i.e. the regions not necessarily contain their corresponding sites; and (iii) it is not clear how to set the weights to achieve the best cell size (taking into account a desired size distribution).

The definition of a *spherical* Laguerre diagram is the same as before but using the *spherical Laguerre distance* which is defined as

$$\tilde{d}_L(P,A_i) = \frac{\cos\tilde{d}(P,A_i)}{\cos w_i} \tag{2.2.3}$$

where $\tilde{d}(P,Q)$ is the geodesic distance (given by arcs of great circles) between the points $P,Q \in \mathbb{S}$ (the unit sphere). In this new setting the weights are restricted to be in the interval $[0, \frac{\pi}{2})$. For more details about this definition and its properties we refer the reader

[1] This is not a proper distance but rather a degree of "farness" [Sugihara, 2002]

to the original paper [Sugihara, 2002]. We discuss next just the main properties that allows us to construct the spherical Laguerre diagram in a similar way as the spherical Voronoi diagram was constructed before in Chapter 1.

First, the boundary between two regions $\mathscr{L}_i, \mathscr{L}_j$ of the spherical Laguerre diagram is an arc of great circle that crosses perpendicularly the geodesic (arc of great circle) joining the corresponding sites A_i, A_j. Note that this arc may be empty if the two regions do not share a boundary. Second, the precise point where the boundary arc meets the geodesic can be determined from w_i, w_j. And third, the transformation $A_i \mapsto \frac{A_i}{\cos w_i}$ mapping $\mathscr{K} = \{A_i\}_{i=1}^n$ into $\mathscr{K}^*$ is such that the dual of the spherical Laguerre diagram associated to $\mathscr{K}$ is equivalent to the convex hull of $\mathscr{K}^*$. Using these properties we can devise an algorithm for computing the spherical Laguerre diagram as done before for spherical Voronoi diagrams: we compute the convex hull of $\mathscr{K}^*$ and reuse that information to reconstruct the boundary of the regions $\{\mathscr{L}_i\}_{i=1}^n$.

Taking into accounts the above we can compute a scaffold using spherical Laguerre diagrams with modified versions of Algorithm 1 and 3 of Section 1.5. We define w_e^v to be the weight associated to the edge $e \in \mathscr{E}_\Sigma,\ e \multimap v$. In the modified version of Algorithm 1 we then take $\mathscr{H}_v \leftarrow \mathit{convexHull}(\mathscr{A}_v^*)$, where $\mathscr{A}_v^* = \{\frac{A_e}{\cos w_e^v} : A_e \in \mathscr{A}_v\}$ is the set of transformed sites in $\mathbb{S}_v$. In Algorithm 3 we need to modify the way the normals of a face are computed. For a face $F \in \mathscr{H}_v$, let $\{A_1, A_2, A_3\} \subset \mathscr{A}_v$ be three consecutive vertices of F in $\mathscr{H}_v$ (i.e. such that A_1A_2 and A_2A_3 are edges of F) with corresponding weights $\{w_1, w_2, w_3\}$ respectively. Then the a normal N to F is defined as

$$N = \frac{(\cos w_2 A_1 - \cos w_1 A_2) \times (\cos w_3 A_2 - \cos w_2 A_3)}{\|(\cos w_2 A_1 - \cos w_1 A_2) \times (\cos w_3 A_2 - \cos w_2 A_3)\|}. \tag{2.2.4}$$

If N is not an outward pointing normal (from the center v of $\mathbb{S}_v$) we take $-N$ as the normal for F. This construction is justified by the following proposition.

Proposition 2.2.1 (Spherical Laguerre diagram construction)**.** *Let $K = \{A_i\}_{i=1}^n \subset \mathbb{S}$ be a set of sites on the unit sphere $\mathbb{S}$, and $\{w_i\}_{i=1}^n \subset [0, \frac{\pi}{2})$ the corresponding weights. Then:*

(i) The boundary between the regions $\mathscr{L}_i$ and $\mathscr{L}_j$ ($i, j \in \{1, 2, \ldots, n\}$ $i \neq j$) is a subset of the great circle defined by the intersection of $\mathbb{S}$ and the plane with unit normal

$$N_{i,j} = \frac{\cos w_j A_i - \cos w_i A_j}{\|\cos w_j A_i - \cos w_i A_j\|}.$$

(ii) The intersection of the boundary arcs separating $\mathscr{L}_i$ from $\mathscr{L}_{j_0}$, and $\mathscr{L}_i$ from $\mathscr{L}_{j_1}$ is a subset of $\{N_{i,j_0,j_1}, -N_{i,j_0,j_1}\}$ with

$$N_{i,j_0,j_1} = \frac{N_{i,j_0} \times N_{i,j_1}}{\|N_{i,j_0} \times N_{i,j_1}\|}.$$

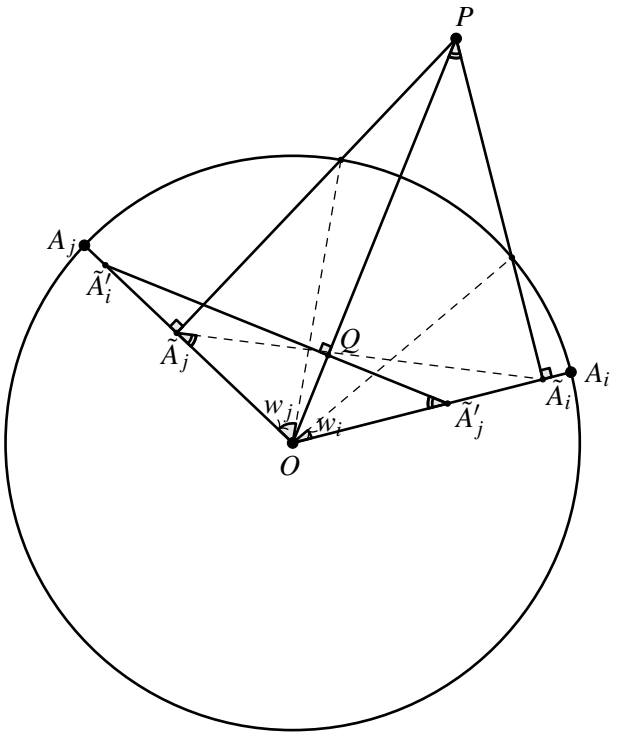


Figure 2.7: $\mathbb{S}$ projected to the plane containing the triangle OA_iA_j. The boundary between the regions $\mathscr{L}_i$ and $\mathscr{L}_j$ is a subset of the great circle supported by the plane containing OP and perpendicular to OA_iA_j.

Proof. In Figure 2.7 we show the projection of $\mathbb{S}$ on the plane containing the triangle OA_iA_j (O is the origin and the center of $\mathbb{S}$). The weights w_k ($k = i, j$) are drawn as angles supporting the half cord passing trough $\tilde{A}_k = \cos w_k A_k$ perpendicularly to OA_k. The intersection of the lines supporting the half cords is denoted as P. In [Sugihara, 2002] it was proved that the boundary between $\mathscr{L}_i$ and $\mathscr{L}_j$ is a subset of the great circle defined by the plane containing OP and perpendicular to the plane containing OA_iA_j. We are going to prove next that $N = \cos w_j A_i - \cos w_i A_j$ is perpendicular to OP, from here it follows (i), i.e. $N_{i,j} = \frac{N}{\|N\|}$ is the unit normal of the plane defining the great circle that supports the boundary between $\mathscr{L}_i, \mathscr{L}_j$. To see that $N \perp OP$ we proceed as follows. Let $\tilde{A}'_i = \cos w_j A_i$, $\tilde{A}'_j = \cos w_i A_j$ and Q be the intersection of OP with $\tilde{A}'_i\tilde{A}'_j$. Then since $O\tilde{A}'_i = O\tilde{A}_i$ and $O\tilde{A}'_j = O\tilde{A}_j$ it follows that $\angle O\tilde{A}'_j\tilde{A}'_i = \angle O\tilde{A}_j\tilde{A}_i = \angle OP\tilde{A}_i$ (because the quadrilateral $O\tilde{A}_jP\tilde{A}_i$ is cyclic). The last equality shows that $Q\tilde{A}'_j\tilde{A}_iP$ is also cyclic, therefore $\angle PQ\tilde{A}'_j = \angle P\tilde{A}_iO = \frac{\pi}{2}$.

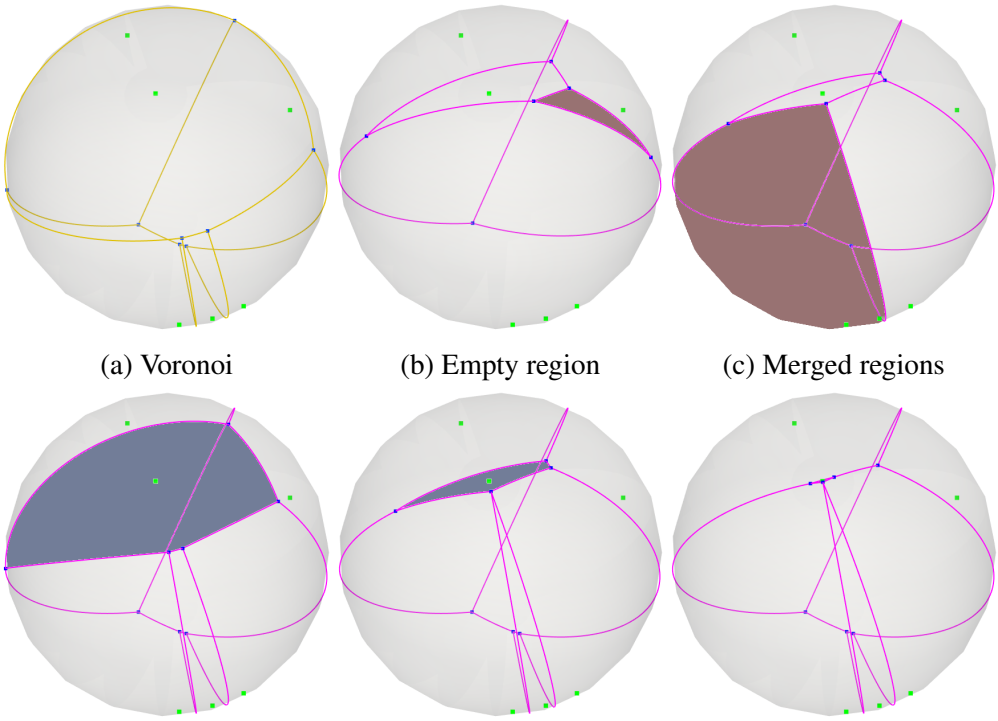
From (i) it follows that the intersection of the two boundary arcs between $\mathscr{L}_i, \mathscr{L}_{j_0}$ and $\mathscr{L}_i, \mathscr{L}_{j_0}$ is perpendicular to both N_{i,j_0} and N_{i,j_1}, thus the intersection is on the line passing trough O with direction $N_{i,j_0} \times N_{i,j_1}$. This proves (ii). □

2.2.2 Limitations

The issues described for planar Laguerre diagram are still open questions in the setting of spherical Laguerre diagrams for scaffolding. In particular it is not clear how to pick the weights w_i to properly reflect the radii distribution and guarantee feasibility. In practice we follow the naive approach of picking $w_i = \frac{r_i}{\max r_i}(\frac{\pi}{2} - \varepsilon)$ where r_i is the radius associated to the i-th incident edge and $\varepsilon > 0$ is a small constant.

This approach does not work in general since it may happen that at least one Laguerre region does not include the corresponding site (this is the case when a region associated to a site is empty, Figure 2.8b). We think that the values of w_i and ε may be picked in an optimization process where the cell sizes approach a desired shape while still guaranteeing feasibility but this needs further research. The influence of the value of ε can be appreciated in Figure 2.8.

Another important issue is that it is not clear yet how to prove feasibility for Laguerre diagrams. The proof of feasibility for Voronoi diagrams relies strongly on the fact that the dual of the Voronoi diagram is the convex hull of points $\mathscr{A}_v$ on the sphere. This is no longer the case with $\mathscr{A}_v^*$ since the convex hull of $\mathscr{A}_v^*$ in general is not equivalent to an inscribed polyhedron.



(a) Voronoi (b) Empty region (c) Merged regions

(d) $\varepsilon = 0.1$ (e) $\varepsilon = 0.01$ (f) $\varepsilon = 0.001$

Figure 2.8: Spherical Laguerre diagram examples. In the top row we show (a) the spherical Voronoi diagram for reference, along with two bad cases of the spherical Laguerre diagrams: (b) when one region does not contain a site, and (c) when some regions are merged into one containing more than one site. In the bottom row we show the effect of varying ε. In the examples all cells have associated radii 1 except the shrinking one that has radii 0.9. We can appreciate how taking small values of ε diminishes the sensibility in the difference between radii, i.e. the cell shrinks more for lower values of ε.

2.3 Conclusions

In this chapter we presented some further topics on scaffolding. We discussed how scaffolds can be used for hexahedral volumetric meshes, and how we can use spherical Laguerre diagrams to obtain better cell shapes. We presented the geometric ideas needed in the development of algorithms with Laguerre diagrams. Hexahedral meshing and Laguerre diagrams can be combined to control the shapes and sizes of the hexahedra in volumetric meshes in accordance with the radii at joints.

We provide working prototypes of the ideas in this chapter but still more research is needed. In the case of hexahedral meshes, an analysis should be performed on the general shapes of the hexahedra and the singularities of the mesh under different subdivision approaches, since this impacts applications (like material simulation). We also pointed out some limitations and paths of research for spherical Laguerre diagrams. Although still limited, these new ideas could lead to a very powerful method with many applications.

Part II

Convolution surfaces

Chapter 3

Introduction to convolution surfaces

Parts of the content of this chapter were adapted from sections of the publication:

[Fuentes Suárez and Hubert, 2018a] Fuentes Suárez, A. J. and Hubert, E. (2018a). Convolution surfaces with varying radius: Formulae for skeletons made of arcs of circles and line segments. In Research in Shape Analysis: WiSH2, Sirince, Turkey, AWM, pages 37–60. Springer.

In this section we present abstract formulas that unify the main definitions of convolution surfaces for 1D or 2D skeletons. They show the main properties of these surfaces stemming from the properties of integration. They also illustrate the meaning of "convolution" in this context. Our contributed mathematical approach complements the more practical approach usually followed in the literature [Blinn, 1982, Bloomenthal and Shoemake, 1991, Hornus et al., 2003, Tai et al., 2004, Hubert and Cani, 2012, Zanni et al., 2013], in Section 3.3 we retake the standard formulas for a more practical discussion.

3.1 Convolution surfaces

Convolution surfaces are the level sets of a convolution field that results from integrating a *kernel* function K along a *skeleton* Σ. A *skeleton* $\Sigma \subset \mathbb{R}^3$ is either

(i) a portion of a regular curve $\Sigma = \Gamma([a,b])$,

(ii) a region of a regular surface $\Sigma = \Pi([a,b],[c,d])$, or

(iii) a finite collection of elements of the type described in (i) or (ii).

The *kernel* is a function $K : \mathbb{R}^+ \to \mathbb{R}^+$ that is at least continuously differentiable (k differentiable in general). The argument is the distance between a point in space and a point on the skeleton. Those kernels are decreasing functions on $\mathbb{R}^+$ and strictly decreasing when non zero: $K(x) > 0 \to K'(x) < 0$.

Once a kernel is chosen the *convolution field* is defined as follows

1. For a skeleton Σ of type (i) or (ii), and a kernel K,

$$C^K_\Sigma(X) = \int_{\mathbb{R}^3} \mathbb{1}_\Sigma(Y) K(\|X-Y\|)\, \mathrm{d}\mathscr{H}^s Y, \tag{3.1.1}$$

   where $\mathscr{H}^s$ denotes the s-dimensional Hausdorff [1] measure with s the dimension of Σ (i.e. $s = 1$ for curves and $s = 2$ for surfaces), and $\mathbb{1}_\Sigma : \mathbb{R}^3 \to \{0,1\}$ is the indicator function of Σ (with $\mathbb{1}_\Sigma(X) = 1 \iff X \in \Sigma$).

2. For $\Sigma = \{\Sigma_j\}_{j=1}^n$ a collection of skeletons of type (i) or (ii), and the real coefficients δ_j,

$$C^\Sigma_K(\mathbf{x}) = \sum_{j=1}^n \delta_j C^{\Sigma_j}_{K_j}(\mathbf{x}). \tag{3.1.2}$$

   Usually $\delta_j = 1$ for all j. We discuss in Chapter 5 how to use other values of these coefficients.

[1] For the main properties of Hausdorff measure we refer to [Evans, 2015, chapters 2 and 3], and [Federer, 1996, sections 2.10 and 3.2].

Finally the *convolution surface* associated to the level value $c \in \mathbb{R}$ ($c > 0$) is defined as

$$\mathscr{S}^K_{\Sigma,c} = \left\{P \in \mathbb{R}^3 \,:\, C^\Sigma_K(P) = c\right\}. \tag{3.1.3}$$

When K, Σ or c are known from the context we may omit them and write, for example, $\mathscr{S}_\Sigma$ or C_Σ.

Equation (3.1.1) can be written as the convolution[2] of $\mathbb{1}_\Sigma$ with $\widetilde{K}(\cdot) = K(|\cdot|)$:

$$(\mathbb{1}_\Sigma * \widehat{K})(X) = \int_{\mathbb{R}^3} \mathbb{1}_\Sigma(Y)\widetilde{K}(Y-X)\,\mathrm{d}\mathscr{H}^s Y.$$

Therefore a convolution surface can be seen as the convolution of the kernel with the skeleton.

As the inverse image of a closed set by a k-continuously differentiable map, $k \geq 1$, the resulting *convolution surfaces* (Equation 3.1.3) are closed (in a topological sense) and smooth, provided c is not a critical value[3] of C^Σ_K. It is the boundary of a smooth 3-dimensional manifold $V_c = \left\{P \in \mathbb{R}^3 : C^\Sigma_K(P) \geq c\right\}$. Furthermore V_{c_0} and V_{c_1} are diffeomorphic provided that there is no critical values in the interval $[c_0, c_1]$ [Milnor, 1963, Theorem 3.1]. In general it is preferred that $\Sigma \subset V_c$. In that way we can guarantee that the surface surrounds the skeleton. This property depends on the kernel. As we discuss later some interesting kernels do not provide a surrounding surface for all values of the level set.

3.2 Discussion on the choice of a kernel

The first convolution surfaces [Bloomenthal and Shoemake, 1991] were based on the Gaussian kernel $x \mapsto e^{-\sigma r^2}$ (also in [Blinn, 1982]) that depends on a parameter $\sigma > 0$. The difficulty in evaluating the resulting convolutions prompted the introduction of kernels that provided closed form expressions for the convolution fields associated to basic skeleton elements. [Sherstyuk, 1999a, Sherstyuk, 1999b] promoted the Cauchy kernel $x \mapsto \frac{1}{(1+\sigma x^2)^2}$ after [Wyvill et al., 1986] introduced the inverse function $x \mapsto \frac{1}{x}$. For faster convolution [Cani and Hornus, 2001, Hornus et al., 2003] introduced the power inverse cube kernel $x \mapsto \frac{1}{x^3}$. [Jin and Tai, 2002a] also exhibited the benefit of using the quintic inverse $x \mapsto \frac{1}{x^5}$. With the power inverse kernels and the Cauchy kernels, the whole skeleton influences the convolution field at a point, even if infinitesimally. In order to

[2] For $f, g : \mathbb{R}^3 \to \overline{\mathbb{R}}$ the convolution of f and g, is defined as $f * g : \mathbb{R}^3 \to \overline{\mathbb{R}}$ with

$$(f*g)(x) = \int_{\mathbb{R}^3} f(y)g(x-y)\,\mathrm{d}y.$$

[3] A critical point of a function $f : (x,y,z) \mapsto f(x,y,z)$ is a point (x_0,y_0,z_0) at which the gradient (f_x, f_y, f_z) of f vanishes. A critical value of f is the value $f(x_0,y_0,z_0)$ of f at a critical point (x_0,y_0,z_0).

limit the zone of influence of each point of the skeleton, compact support kernels were introduced, mostly as piecewise polynomial functions. To some extent, they allow to limit the unwanted bulges or blending (Figure 3.1). Their use nonetheless necessitates to determine the geometry of the intersection of the skeleton with spheres in order to use symbolic formulas.

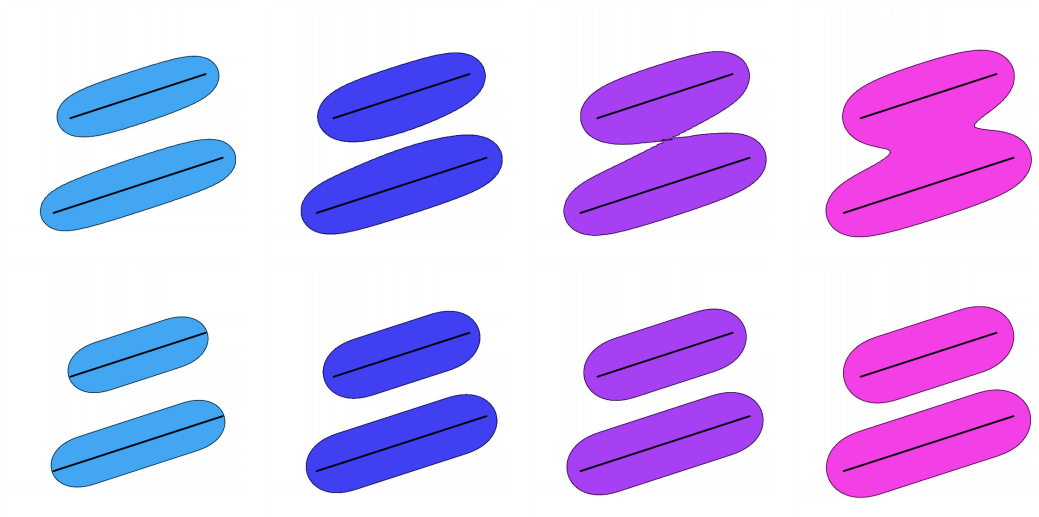


Figure 3.1: Convolution curves based on two parallel segments. The top line uses the cubic inverse kernel and the bottom line a compact support kernel. Columns correspond to an identical sought thickness. For the cubic inverse kernel, a bulge and then a blend appear between the line segments, while the convolution surfaces get only thicker with the compact support kernel.

Kernels may be grouped into families with elements indexed by the degree of smoothness. Therefore, the family of power inverse kernels is given by

$$K_i^{inverse} : x \mapsto \left(\frac{1}{r}\right)^i$$

for i any positive integer. Similarly, the Cauchy family of kernels is given by

$$K_i^{Cauchy} : x \mapsto \left(\frac{1}{1+\sigma r^2}\right)^{\frac{i}{2}},$$

and the family of compact support polynomial kernels used in [Hubert, 2012] is defined as

$$K_i^{compact} : x \mapsto \begin{cases} \left(1-\frac{x^2}{R^2}\right)^{\frac{i}{2}} & \text{if } r < R \\ 0 & \text{otherwise.} \end{cases}$$

The kernel function (power inverse, Cauchy, compact support, ...) may be selected so as to have closed form expressions for the convolution fields associated to basic

skeleton elements (line segments, triangles, ...). The additivity property of integration makes the convolution field independent of the partition of the skeleton. General skeletal curves are partitioned and approximated by a set of basic elements. The convolution field for the whole skeleton is obtained by adding the convolution fields of the constitutive basic elements. See for instance [Bloomenthal and Shoemake, 1991, Cani and Hornus, 2001, Hornus et al., 2003, Jin and Tai, 2002a, Jin and Tai, 2002b, Jin et al., 2001, Sherstyuk, 1999a, Sherstyuk, 1999b, Zanni, 2013, Zanni et al., 2013].

The convolution surfaces obtained with a power inverse kernel always enclose the skeleton since these functions tend to infinity when approaching the skeleton. As exploited in [Hubert, 2012, Hubert and Cani, 2012] the closed form formulas of convolution fields using the family of Cauchy kernels differ only slightly from the ones using power inverse kernels. In the case of compact support polynomial kernels, one considers only K_i^{comp} for $i \geq 3$. This is to guarantee that the resulting convolution surfaces are at least continuously differentiable. For $i < 3$, K_i^{comp} is not differentiable at $x = R$. The case $i = 4$ is actually the case considered in [Jin et al., 2008, Sherstyuk, 1999b]. In all the families if we increase i we obtain smoother shapes. Figure 3.2 show the typical graphs of the power inverse and compact support polynomial families.

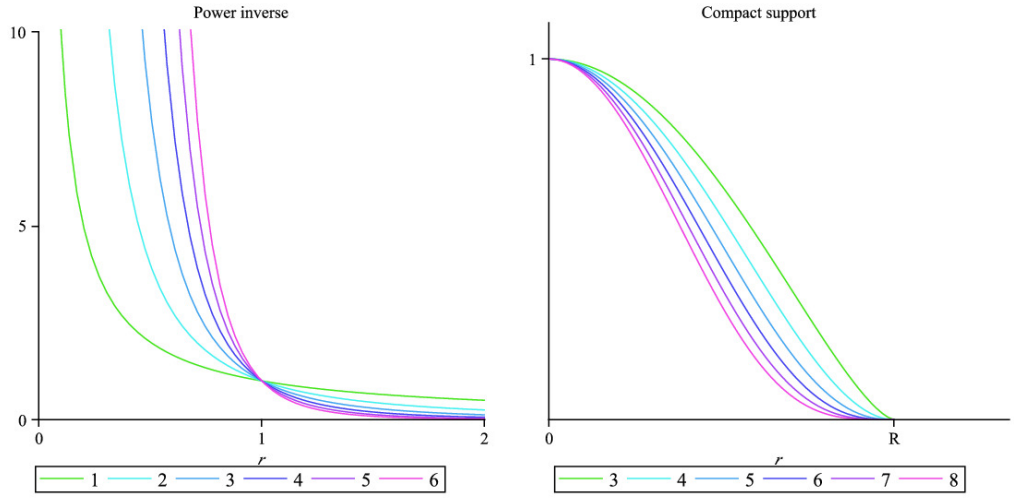


Figure 3.2: The graphs of the kernel functions $K_i^{inverse}$ and $K_i^{compact}$, varying i.

With compact support kernels $K_i^{compact}$ the smoothness of the convolution surface increases with i. With power inverse kernels $K_i^{inverse}$ the convolution fields are smooth at all points outside the skeleton. Yet, as i increases, the convolution surface is *sharper* around the skeleton. This is illustrated in Figure 3.3.

When the convolution field has a critical point, chances are that there is a change in the topology of the convolution surface as it goes through the critical value [Milnor, 1963]. This is illustrated in Figure 3.1 with a skeleton made of two line segments. The con-

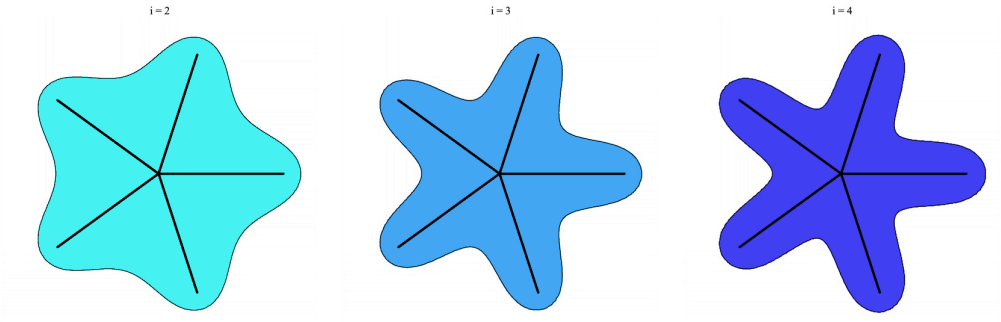


Figure 3.3: Convolution curves for a set of segments with power inverse kernels of degree 2, 3 and 4. The level set was chosen as to have identical thickness at the tips. Note that sharpness increases from left to right.

volution field has a critical point and the convolution surface through this point has a singularity. The corresponding level set is a transition from bulging to blending, from two connected components to a single component. Figure 3.1 also illustrates the fact that compact support kernels allow to dismiss the influence of skeleton elements that are at distance more than R, thus avoiding some of the bulging and blending that appear for the kernels with infinite support kernel.

3.3 Convolution of regular curves

Consider a regular curve $\Gamma : [a,b] \subset \mathbb{R} \to \mathbb{R}^3$. This means that Γ is continuously differentiable and Γ' does not vanish in $[a,b]$. The infinitesimal arc-length is then $\|\Gamma'(t)\|\mathrm{d}t$. When Γ serves as skeleton, that is $\Gamma \in \Sigma$, we call Γ *skeletal curve*. The *convolution field* $C^K_\Gamma : \mathbb{R}^3 \to \mathbb{R}$ associated to Γ described in (3.1.1) can be written as

$$C^K_\Gamma(X) = \int_a^b K\left(\|X - \Gamma(t)\|\right) \|\Gamma'(t)\| \mathrm{d}t. \tag{3.3.1}$$

When $\Gamma : [0,l] \to \mathbb{R}^3$ is parameterized by arc length s, Equation (3.3.1) becomes

$$C^K_\Gamma(X) = \int_0^l K\left(\|X - \Gamma(s)\|\right) \mathrm{d}s. \tag{3.3.2}$$

The integral is independent of the (regular) parametrization of the curve used. convolution fields with a power inverse kernel are infinitely differentiable outside of the curve $\Gamma([a,b])$. convolution fields with a compact support kernel $K = K_i^{compact}$ are $\lfloor \frac{i-1}{2} \rfloor$-continuously differentiable.

3.4 Line segments and arcs of circle

In practice the most commonly used skeletal curves are line segments, arcs of circle, or low degree Bezier curves. The widely used approach for more complex skeletal curves is to use an approximation by line segments [Cani and Hornus, 2001, Hornus et al., 2003, Jin et al., 2001, Sherstyuk, 1999a, Zanni et al., 2013]. Some issues with this versatile approach is that either the resulting convolution surface presents some visible turns at the joints of line segments, or the number of segments must be increased significantly in order to get a visually smooth surface. Convolution for arcs of circles were also examined in [Jin and Tai, 2002b, Zanni et al., 2011]. [Jin and Tai, 2002b] dealt with planar skeletons. [Zanni et al., 2011] was geared towards skeleton curves with nonzero torsion: arcs of circles, and line segments, were deformed into helices that have powerful modeling properties for natural shapes, in particular for the animation of hair [Bertails et al., 2006]. The warping technique used in [Zanni et al., 2011] allows to decrease the number of skeleton basic elements to be used to obtain a natural looking shape. This provides a substantial gain on the computational cost as meshing the surface requires the repeated evaluation of the convolution function. Yet the surfaces obtained by warping in [Zanni et al., 2011] exhibit artifacts and singularities so that this technique requires a fine tuning of the warping parameters.

An alternative approach is to exploit the capabilities of curves made of line segments and arcs of circles to provide a $\mathscr{G}^1$ approximation of the skeletal curve. Arcs of circles have the great advantage to allow the construction of $\mathscr{G}^1$-curves that can approximate any curve [Nutbourne and Martin, 1988, Song et al., 2009]. One can thus achieve both mathematically smooth and visually appealing shapes with skeleton curves consisting of few basic elements.

3.5 Varying thickness

Several alternatives have been introduced for varying the thickness of the convolution surface along the skeleton. A first idea was to use a *weight* function along the skeleton [Bloomenthal and Shoemake, 1991]. For a skeleton given by a regular curve $\Gamma : [a,b] \to \mathbb{R}^3$, one uses a weight function $w : [a,b] \to \mathbb{R}$. The convolution field is then defined as

$$J^K_{\Gamma,w}(X) = \int_a^b w(t) K\left(\|X - \Gamma(t)\|\right) \|\Gamma'(t)\| \mathrm{d}t. \tag{3.5.1}$$

In this formulation the convolution field is now dependent on the parametrization used for the skeleton curve.

Polynomial weights were studied in [Hubert and Cani, 2012, Jin and Tai, 2002a, Jin and Tai, 2002b, Jin et al., 2001], where closed-form formulas was obtained for particular cases of weighted convolution. The drawback of this approach was illustrated

in [Hubert and Cani, 2012, Figure 9]: the influence of the weight diminishes as the degree of the kernel increases. Alternative more intrinsic formulations were proposed in [Hornus et al., 2003, Zanni et al., 2013].

Varying radius as proposed by [Hornus et al., 2003]

In [Hornus et al., 2003] Hornus *et al.* proposed a different way of computing the convolution field. This allows the user to modify the shape of the final surface according to a radius assigned to each point in the skeleton: the distance is divided by the radius at the corresponding point in the skeleton. A radius is given by a function $\rho : [a,b] \to \mathbb{R}^+$. The convolution field is then defined by

$$H^K_{\Gamma,\rho}(X) = \int_a^b K\left(\frac{\|X-\Gamma(t)\|}{\rho(t)}\right)\|\Gamma'(t)\|\mathrm{d}t. \tag{3.5.2}$$

If additional precaution are not imposed on the radius function ρ, this convolution field depends on the parametrization of the curve Γ. In practice it makes sense to have a radius function that is approximately linear in the arc-length.

SCALIS as proposed by [Zanni et al., 2013]

An alternative convolution introduced in [Zanni et al., 2013] allows to properly model shapes with pieces at different scales. This lead to the name SCALe invariant Integral Surfaces (SCALIS). In this case the convolution field is

$$Z^K_{\Gamma,\lambda}(X) = \int_a^b K\left(\frac{\|X-\Gamma(t)\|}{\lambda(t)}\right)\frac{\|\Gamma'(t)\|}{\lambda(t)}\mathrm{d}t, \tag{3.5.3}$$

where $\lambda : [a,b] \to \mathbb{R}^+$ is called the *scale function*. Notice that λ plays a similar role as ρ in the formulation by Hornus. To apprehend all the good features of this new definition of convolution surface, the reader is referred to [Zanni, 2013, Zanni et al., 2013]. Let us just observe the case where λ is a constant. If we write $\lambda \cdot P$ and $\lambda \cdot \Gamma$ for the point and the regular curve obtained through a homothety (*a.k.a* homogeneous dilatation or scaling) with ratio λ, then

$$\mathscr{S}^K_{\lambda\cdot\Gamma,\lambda}(\lambda\cdot P) = \mathscr{C}^K_\Gamma(P) \quad \text{or equivalently} \quad \mathscr{S}^K_{\Gamma,\lambda}(P) = \mathscr{C}^K_{\lambda^{-1}\cdot\Gamma}\left(\lambda^{-1}\cdot P\right). \tag{3.5.4}$$

This implies that the convolution surface of equation $\mathscr{S}^K_{\lambda\cdot\Gamma,\lambda}(P) = c$ is homothetic to the convolution surface $\mathscr{C}^K_\Gamma(P) = c$ with a ratio λ. This is an important property of SCALIS that allows to model at different scales, and hence is called *scale invariance*.

3.6 Closed form formulas

Closed-form formulas for convolution fields have been studied for several skeleton primitives and kernels [Hubert, 2012, Zanni et al., 2011, Hubert and Cani, 2012, Jin and Tai, 2002a, Jin and Tai, 2002b, Jin et al., 2001, Sherstyuk, 1999b, Zanni, 2013]. Given an appropriate parametrization of a skeleton primitive and a kernel, the closed form formulas for the convolution field is very often obtained thanks to symbolic integration [Bronstein, 2005], which is implemented in computer algebra software as Maple or Mathematica. The field function for the pair consisting of the skeleton primitive and kernel can then be implemented with a view on optimizing its evaluation cost. As different kernels change the properties of the convolution surface, even if lightly, it is interesting to offer, in the same geometric modeling software, alternative kernels for convolution. The same is true for skeleton primitives. As different kernels may bring related closed form formulas one seeks to optimize the code by taking advantage of the common subexpressions [Jin et al., 2001, Equations 11-14]. A greater level of generality was offered with the new approach in [Hubert, 2012, Hubert and Cani, 2012] where common subexpressions were encapsulated into low order recurrence formulas: kernels were grouped into families, each indexed by an integer, and, for each family, recurrence relationships were exhibited for the convolution function field, indexed by the same integer. The recurrence relationships that appeared in [Hubert, 2012, Hubert and Cani, 2012] mostly came from tables of known integrals, which made this strategy uneasy to generalize.

We discuss later in Chapter 4 a technique that allows to obtain algorithmically the recurrence formulas on the convolution fields for families of kernels. This approach relies on Computer Algebra Systems and Creative Telescoping [Chen and Kauers, 2017, Chyzak and Salvy, 1998, Koutschan, 2013]. The formulas so generated can be translated into optimized C code, as illustrated in [Hubert and Cani, 2012]. This approach should be balanced against the use of highly accurate numerical integration routines based on quadratures [Piessens et al., 1983]. Indeed, some closed form formulas might be judged too complex to result in efficient code for evaluating the convolution functions.

3.7 Numerical integration

Numerical integration is the alternative approach for the computation of the convolution fields. The formulas involved are too complex for ad-hoc implementations of integration by quadrature methods (see for example [Fuentes Suárez and Hubert, 2018a]). This is one of the reasons why numerical integration has been mostly ignored in the convolution surfaces context. The speed of computations is also limiting in the context of real-time applications (like interactive modeling) where alternative real-time intermediate representations should be used.

An effective approach for the complex integral formulas is to use QUAD-

PACK [Piessens et al., 1983]. This collections of algorithms can handle integrals with singularities and improper integrals. In this work we refer to the QUADPACK implementation in the Gnu Scientific Library [Galassi et al., 2017].

The main machinery of the numerical integration in QUADPACK/GSL is the Gauss-Kronrod quadrature formulas [Burden and Faires, 2011, Notaris, 2016, Trefethen, 2008]. The Gauss quadrature method allows to approximate an integral by summing up the values of the integrand at *quadrature points*. Given the quadrature point $\{x_i\}_{i=1}^{n} \subset [-1,1]$ one can write $\int_{-1}^{1} f(t)dt = \sum_{i=1}^{n} c_i f(x_i) + R_n(f)$, where the coefficients $c_i \in \mathbb{R}$ and points x_i depend on n, and $R_n(f)$ is the error of the approximation. The n-points Gauss quadrature method computes exactly (i.e. $R_n(f) = 0$) the integral of all polynomials of degree lower than $2n$. Gauss-Kronrod method improves upon this by increasing the degree of the exactly computed polynomials up to $3n+1$ [Notaris, 2016]. For general functions there are absolute and relative error bounds for $R_n(f)$. This method is not limited to integrals on $[-1,1]$, there are techniques to transform this interval into the actual interval of integration.

The family of QUADPACK algorithms has also specialized algorithms for adaptive integration where the computations are made finer (i.e. more quadrature points are added by splitting the interval of integration) when some error threshold is not attained (adaptivity). In GSL library we can pick the number of quadrature points to be used. In the context of convolution one can use the 61 points Gauss-Kronrod rule (maximum in GSL). Another interesting feature of GSL/QUADPACK is that we can specify in advance the singular points in the integrand and get better and faster approximations.

Chapter 4

Closed form formulas from recurrence

The content of this chapter was adapted from sections of the publication:

[Fuentes Suárez and Hubert, 2018a] Fuentes Suárez, A. J. and Hubert, E. (2018a). Convolution surfaces with varying radius: Formulae for skeletons made of arcs of circles and line segments. In Research in Shape Analysis: WiSH2, Sirince, Turkey, AWM, pages 37–60. Springer.

We present here a technique that allows to obtain algorithmically the recurrence formulas on the convolution fields for families of kernels and a given skeleton primitive. This unified approach relies on Creative Telescoping which is used here in the context of convolution surfaces.

In this section we give a brief introduction to *Creative Telescoping* (CT) and illustrate how to use some specialized software for our purpose. In Section 4.2 and Section 4.3 we show some specific recurrences we obtained with CT for line segments and arcs of circle with power inverse and compact support kernels. All the recurrences we provide, as well as the recurrence equations, can be checked by a simple differentiation.

4.1 Creative Telescoping

We introduce here Creative Telescoping. This is an active research field [Chen and Kauers, 2017], with new algorithms and new applications appearing every year. For a complete (but still gentle) introduction we refer the reader to [Koutschan, 2013], and to [Chyzak and Salvy, 1998] for a formal development of the ideas we present here.

Creative Telescoping comes in several guises depending on the application. We want to find recurrence relations for the integrals in the convolution field. For this case CT works with *differential* (D_x) and *shift* (S_n) operators. These operators act by differentiating or incrementing by one (in the respective variable) the input expression f, i.e. $D_x f(x,n) = \frac{\partial}{\partial x} f(x,n)$ and $S_n f(x,n) = f(x,n+1)$. Here x and n stand for continuous and discrete variables respectively.

Higher order linear operators are constructed as elements of the vector space $\mathbb{O}$ spanned by the symbols of the form $D_x^{\alpha_1} D_y^{\alpha_2} \dots S_n^{\beta_1} S_m^{\beta_2} \dots$ with coefficients in $\mathbb{F} = \mathbb{K}(x,y,\dots,n,m,\dots)$, the field of rational functions over a field $\mathbb{K}$ (of characteristic 0). $\mathbb{O}$ is actually a $\mathbb{F}$-algebra $\mathbb{F}\langle D_x, D_y, \dots, S_n, S_m\rangle$ in which the generators $(D_x, D_y, \dots, S_n, S_m)$ commute pairwise (for instance $D_x S_n = S_n D_x$) but the commutation of an operator with an element in the field is not trivial:

$$D_x p = p D_x + \frac{\partial}{\partial x} p \quad \text{and} \quad S_n p = p|_{n \to n+1} S_n.$$

$\mathbb{O}$ is a so called Ore algebra. Ore algebras are defined for operators that generalize both derivations and shifts [Ore, 1933]. In this context one introduces the concept of ∂-finite functions [Chyzak and Salvy, 1998]. These are often called *holonomic* functions though there are subtleties between the two notions. Creative Telescoping takes ∂-finite functions as input. Basically, a ∂-finite function f is uniquely and well defined as the zero of a set of linear operators in $\mathbb{O}$ with prescribed values for a finite subset of its low order derivatives or shifts (the *initial conditions*). It is then the case that for any operator

∂ (of the form D_x or S_n) there is an integer m such that $\{\partial^k f \mid k = 0..m\}$ are linearly dependent over $\mathbb{F}$.

For illustration consider

$$f(t,i,k) = \frac{(\lambda+\delta t)^k}{(at^2-2bt+c)^i}, \text{ with } a,b,c,\delta,\lambda \in \mathbb{R}.$$

For the operators D_t, S_i and S_k with $\mathbb{F} = \mathbb{R}(t,i,k)$, we see that f is ∂-finite by observing that

$$S_k f = \frac{(\lambda+\delta t)^{k+1}}{(at^2-2bt+c)^i} = (\lambda+\delta t)\, f, \quad S_i f = \frac{(\lambda+\delta t)^k}{(at^2-2bt+c)^{i+1}} = \frac{1}{at^2-2bt+c} f,$$

and

$$D_t f = \left(\frac{\delta k}{\lambda+\delta t} - 2i \frac{at-b}{at^2-2bt+c} \right) f.$$

Given a complete set of operators annihilating the function, the purpose of Creative Telescoping is to produce operators $P \in \mathbb{O}$ such that

$$P = \mathscr{L} - D_t C \quad \text{and} \quad Pf = 0,$$

with $\mathscr{L}, C \in \mathbb{O}$ and where $\mathscr{L}$ depends neither on D_t nor on t. That is $\mathscr{L}$ is a linear combination of $S_i^{\beta_1} D_k^{\beta_2}$ over $\mathbb{K}(i,k)$. Because of its total independence on t, $\mathscr{L}$ commutes with an integration operator with respect to t so that

$$0 = \int_{t_0}^{t_1} P f dt = \int_{t_0}^{t_1} \mathscr{L} f dt - \int_{t_0}^{t_1} D_t C f dt = \mathscr{L} \left(\int_{t_0}^{t_1} f dt \right) - [Cf]_{t=t_0}^{t=t_1}.$$

Thus $F(i,k) = \int_{t_0}^{t_1} f(t,i,k) dt$ satisfies the recurrence $\mathscr{L} F(i,k) = g(t_1) - g(t_0)$, where $g = Cf$. The operator $\mathscr{L}$ is called the *telescoper* and C the *certificate*. We thus have obtained a recurrence relationship on the functions $F(i,k) = \int_{t_0}^{t_1} f(t,i,k)\, dt$. For a given skeleton primitive and a family of kernels, the convolution fields are functions of this type. Creative Telescoping thus provides a general unified approach to obtain the recurrence formulas on these.

Note that Creative Telescoping refers to a panel of algorithms. There is no canonical output. For instance one might trade a low order telescoper for a telescoper with lower degree coefficient [Chen and Kauers, 2012], or a telescoper computed with a more efficient heuristic [Koutschan, 2010]. Furthermore, specialized lower complexity algorithms exist for subclasses of functions, like rational, hypergeometric or hyperexponential functions [Bostan et al., 2013, Bostan et al., 2016]. The function f we started with is actually *hypergeometric* since

$$\frac{D_t f}{f} \in \mathbb{F}, \quad \frac{S_i f}{f} \in \mathbb{F}, \quad \text{and} \quad \frac{S_k f}{f} \in \mathbb{F}.$$

4.1.1 Practical use

There are several implementations of CT available, a (perhaps limited) set is `Mgfun`[1] by F. Chyzak (in Maple), `HolonomicFunctions`[2] by C. Koutschan (in Mathematica), and `MixedCT`[3] by L. Dumont (in Maple). The description of the algorithms implemented are in the respective publications [Chyzak and Salvy, 1998, Koutschan, 2010, Bostan et al., 2016]. Here, as Maple users, we describe an example of use with `Mgfun`, following up on the running example of previous paragraph.

The operators D_t, S_i, S_k are expressed in `Mgfun` by the `t::diff`, `i::shift`, and `k::shift` directives. In order to get recurrence formulas for the integral

$$F(i,k) = \int_{t_0}^{t_1} f(t,i,k)dt = \int_{t_0}^{t_1} \frac{(\lambda+\delta t)^k}{(at^2-2bt+c)^i}dt,$$

one simply calls the command `creative_telescoping(f(t,i,k), [k::shift,i::shift], t::diff)`. It is indeed practical that an appropriate system of linear operators annihilating f is computed internally. The class of expressions for which this works is given by the closure properties of ∂-finite functions [Chyzak and Salvy, 1998] and accounted for in the documentation of the command `dfinite_expr_to_sys` in the `Mgfun` package.

The command `creative_telescoping(f(t,i,k), [k::shift,i::shift], t::diff)` outputs two pairs $(\mathscr{L},C)$. To interpret the output of `creative_telescoping` the reader must refer to `Mgfun` documentation. The first pair consists of

$$\begin{aligned}\mathscr{L}_1 \;=\;& 2i\left(ac-b^2\right)\left(\lambda^2 a+2\lambda b\delta+\delta^2 c\right)S_i - a\left(2i-2-k\right)\left(a\lambda+\delta b\right)S_k \\ &+\left(2b^2\left(i-k-1\right)-ac\left(2i-k-1\right)\right)\delta^2 - 2ab\lambda\,\left(k+1\right)\delta - a^2\lambda^2\left(k+1\right)\end{aligned}$$

and

$$C_1 = \left(\lambda+\delta t\right)\left(abt-2b^2+ac\right)\delta + a\lambda\,\left(at-b\right),$$

while the second pair consists of

$$\mathscr{L}_2 = a(k-2i+3)S_k^2 + 2(i-k-2)(a\lambda+b\delta)S_k + (k+1)(a\lambda^2+2b\delta\lambda+\delta^2 c)$$

and

$$C_2 = -\delta\,\left(\lambda+\delta t\right)\left(at^2-2bt+c\right).$$

This latter, for instance, translates to the following recurrence, which will be used in Section 4.2:

[1] `https://specfun.inria.fr/chyzak/mgfun.html`

[2] `http://www.risc.jku.at/research/combinat/software/ergosum/RISC/HolonomicFunctions.html`

[3] `http://mixedct.gforge.inria.fr`

$$a(k-2i+3)F(i,k+2)+2(i-k-2)(a\lambda+b\delta)F(i,k+1)+$$
$$+(k+1)(a\lambda^2+2b\delta\lambda+\delta^2c)F(i,k)=\left[\frac{-\delta(\delta t+\lambda)^{k+1}}{(at^2-2bt+c)^{i-1}}\right]_{t_0}^{t_1}.$$

Note that this latter telescoper involves S_k and not S_i. The set of pairs $\{(\mathscr{L}_1,C_1),(\mathscr{L}_1,C_2)\}$ is indeed *minimal* in a sense that we do not wish to make precise here but that depends on the order of k and i in the input. To give a sense of this order, observe that the output of `creative_telescoping(f(t,i,k), [i::shift,k::shift], t::diff)` Also consists of two pairs $(\mathscr{L},C)$, but the second one involves solely S_i.

4.2 Convolution with line segments

We examine the convolution of line segments for power inverse kernels, with varying radius or scale[4]. First we express the convolution functions, with varying radius or scale, in terms of an integral function indexed by two integers:

$$I_{i,k}(a,b,c,\lambda,\delta)=\int_{-1}^{1}\frac{(\lambda+\delta t)^k}{(at^2-2bt+c)^i}dt$$

We then provide recurrence formulas on this integral so as to have all the convolution functions for line segments with (even) power inverse kernels. The recurrence relationships we exhibit can be adapted to work for all powers. Furthermore, though we do not give details, these recurence also allow to deal with the convolution of line segments with the family of compact support kernels, by taking i to be a negative integer. We choose to restrict here to even powers as they provide easier formulas to evaluate (odd power inverse kernels bring out elliptic functions in the convolution of arcs of circles and planar polygons). This does not affect too much the variety of shapes we can obtain.

4.2.1 Integrals for convolution

Two points $A,B\in\mathbb{R}^3$ define the line segment $[AB]$. A regular parametrization for this line segment is given by $\Gamma:[-1,1]\to\mathbb{R}^3$ with $\Gamma(t)=\frac{A+B}{2}+\frac{B-A}{2}t$. Therefore for a point $P\in\mathbb{R}^3$ we have

$$4\,|P\Gamma(t)|^2=|AB|^2t^2-2\overrightarrow{AB}\cdot\overrightarrow{CP}t+|CP|^2 \text{ where } C=\frac{A+B}{2}$$

is the mid point of the line segment $[AB]$. Hence $|\Gamma'(t)|=\frac{|AB|}{2}$.

[4]Results for convolution of weighted line segments with power inverse and Cauchy kernels can be found in [Hubert and Cani, 2012].

The simple convolution of this line segment with the power inverse kernel $K_{2i}^{inverse}$ is thus given by:

$$\mathscr{C}^{2i}_{[AB]}(P) = \frac{|AB|}{2}\int_{-1}^{1}\frac{1}{|P\Gamma(t)|^{2i}}\,\mathrm{d}t = \frac{|AB|}{2}I_{i,0}\left(\frac{1}{4}|AB|^2,\frac{1}{4}\overrightarrow{AB}\cdot\overrightarrow{CP},\frac{1}{4}|CP|^2,\lambda,\delta\right).$$

If we choose the radius function $\rho : [a,b] \to \mathbb{R}$ to be linear in the arclength we can find $\lambda,\delta \in \mathbb{R}$ such that $\rho(t) = \lambda + \delta t$. Convolution with varying radius is then given by:

$$\mathscr{H}^{2i}_{[AB],\rho}(P) = \int_{-1}^{1}\frac{(\lambda+\delta t)^{2i}}{|P\Gamma(t)|^{2i}}\frac{|AB|}{2}\mathrm{d}t = \frac{|AB|}{2}I_{i,2i}\left(\frac{1}{4}|AB|^2,\frac{1}{4}\overrightarrow{AB}\cdot\overrightarrow{CP},\frac{1}{4}|CP|^2,\lambda,\delta\right).$$

If we now take the scale function to be $\Lambda(t) = \lambda + \delta t$, then

$$\mathscr{S}^{2i}_{[AB],\Lambda}(P) = \int_{-1}^{1}\frac{(\lambda+\delta t)^{2i}}{|P\Gamma(t)|^{2i}}\frac{|AB|}{2(\lambda+\delta t)}\mathrm{d}t = \frac{|AB|}{2}I_{i,2i-1}\left(\frac{1}{4}|AB|^2,\frac{1}{4}\overrightarrow{AB}\cdot\overrightarrow{CP},\frac{1}{4}|CP|^2,\lambda,\delta\right).$$

4.2.2 Closed forms through recurrence formulas

First of all, given that

$$I_{1,0}(a,b,c,\lambda,\delta) = \frac{1}{\sqrt{ac-b^2}}\left[\arctan\left(\frac{at-b}{\sqrt{ac-b^2}}\right)\right]_{-1}^{1}$$

we can determine $I_{i,0}(a,b,c,\lambda,\delta)$ for all $i \geq 1$ thanks to the recurrence relationship

$$2i\left(ac-b^2\right)I_{i+1,0} + (1-2i)\,aI_{i,0} = \left[\frac{at-b}{(at^2-2bt+c)^i}\right]_{-1}^{1}.$$

One then observes that:

$$2a(1-i)I_{i,1} = 2(3-i)(a\lambda+b\delta)I_{i,0} - \left[\frac{\delta}{(at^2-2bt+c)^{i-1}}\right]_{-1}^{1}.$$

This recurrence is actually obtained by specializing the following recurrence to $k=-1$

$$\begin{aligned}a(k-2i+3)I_{i,k+2} + 2(i-2-k)(a\lambda+b\delta)I_{i,k+1} + \\ +(k+1)\left(a\lambda^2+c\delta^2+2b\lambda\,\delta\right)I_{i,k} = \left[\frac{-\delta\,(\lambda+\delta t)^{k+1}}{(at^2-2bt+c)^{i-1}}\right]_{-1}^{1}.\end{aligned}$$

One can thus determine $I_{i,k}$ for all $i \geq 1$ and $k \geq 0$ and therefore $I_{i,2i}$ and $I_{i,2i-1}$ that are needed for convolution with varying radius or scale.

Alternatively, to determine the convolution with varying radius, we can consider the recurrence

$$\begin{aligned} &2i(i+1)a\left(ac-b^2\right)I_{i+2,2i+4}+\delta^2(i+1)(1+2i)\left(a\lambda^2+c\delta^2+2b\lambda\,\delta\right)I_{i,2i} \\ &\quad -i\left(\lambda\,a(1+2i)(2b\delta+a\lambda)+\left((4i+5)ca-2(i+2)b^2\right)\delta^2\right)I_{i+1,2i+2} \\ &\qquad\qquad =\left[\frac{(\lambda+\delta t)^{2i+1}}{(at^2-2bt+c)^{i+1}}C\right]_{-1}^{1}. \end{aligned}$$

where

$$\begin{aligned} C \;=\;& \left(\left(ac+2b^2i\right)t^2-bc(3i+2)t+c^2(i+1)\right)\delta^3 \\ &+\left(2ab(1+2i)t^2-\left(2(i+2)b^2+3aic\right)t+bc(i+2)\right)\lambda\,\delta^2 \\ &+\left(a^2(1+2i)t^2-ab(i+2)t-ac(i-1)\right)\lambda^2\delta+ai(at-b)\lambda^3. \end{aligned}$$

A similar recurrence can be obtained for $I_{i,2i-1}$. As the previous ones, it is obtained by Creative Telescoping (see Section 4.1).

4.3 Convolution with arcs of circle

We examine the convolution of an arc of circle for power inverse kernels. Though arcs of circles appear in the literature about convolution surfaces [Jin and Tai, 2002b, Zanni et al., 2011], there is no general formulas for these. In this section we choose a parametrization for arcs of circle that allows us to write the convolution functions (with varying radius or scale) in terms of an integral indexed by two integers:

$$F_{i,k}(a,b,c,\lambda,\delta)=\int_{-T}^{T}\frac{(\lambda+\delta t)^k(t^2+1)^{i-1}}{(at^2-2bt+c)^i}dt$$

We then show how to determine closed form formulas for these integrals thanks to some recurrences. We restrict our attention to convolution of arcs of circle with even power inverse kernels. With odd power inverse kernels, the closed form formulas for the convolution function involve elliptic functions and can be rather impractical to evaluate.

The closed form formulas for the convolution of arcs of circle with the family of compact support kernels are challenging to obtain. The software `MixedCT`[5] by L. Dumont (in Maple) does meet this challenge. The result would be too cumbersome to be presented here and it is not clear at this stage how to use it efficiently.

[5] `http://mixedct.gforge.inria.fr`

4.3.1 Rational parametrization

When it comes to integration, rational functions are the dependable class [Bronstein, 2005]. The main ingredient in obtaining closed-form convolution functions for arcs of circle is to introduce an appropriate rational parametrization.

We assume that the points O, A and B are not aligned and such that $|OA| = |OB| = r$. They define a plane in space and two arcs of circle, one of angle α the other of angle $\pi + \alpha$ for some $0 < \alpha < \pi$. We have

$$\alpha = \arccos\left(\frac{\overrightarrow{OA} \cdot \overrightarrow{OB}}{r^2}\right) \qquad \text{with} \quad 0 < \alpha < \pi$$

and accordingly to which angle is dealt with we set

$$T = \tan\left(\frac{\alpha}{4}\right) \quad \text{or} \quad T = \tan\left(\frac{\pi + \alpha}{4}\right).$$

Momentarily we consider the coordinate system (x, y, z) where the origin is the center of the circle, the x-axis is the bisector of the chosen angle defined by O, A and B and the (x, y) plane is the plane of the circle. See Figure 4.1.

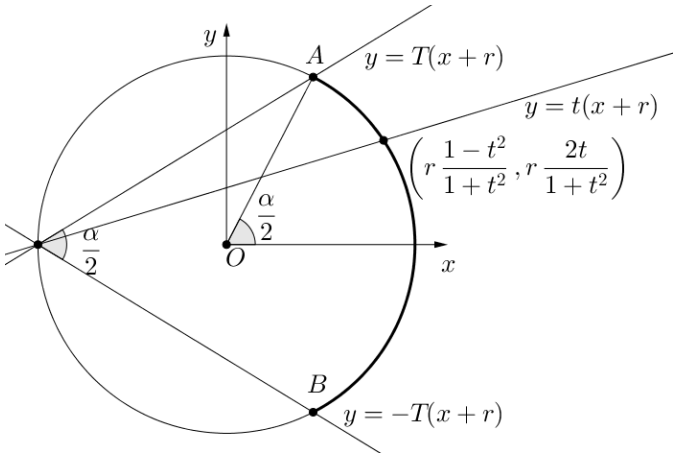


Figure 4.1: Rational parametrization of an arc of circle.

A parametrization of the arc of circle is then given by

$$\begin{array}{rccl} \Gamma: & [-T, T] & \longrightarrow & \mathbb{R}^3 \\ & t & \mapsto & \left(r\dfrac{1-t^2}{t^2+1}, r\dfrac{2t}{t^2+1}, 0\right). \end{array}$$

This is obtained by determining the intersection of the circle with the lines of slope t through the point diametrically opposite to the middle of the arc. Consider a point

$P=(x,y,z)$ in space. We have

$$|P\Gamma(t)|^2=\frac{\alpha t^2-2\beta t+\gamma}{t^2+1} \quad \text{where} \quad \alpha=(x+r)^2+y^2+z^2, \quad \beta=2ry, \quad \gamma=(x-r)^2+y^2+z^2.$$

Note that

$$\begin{aligned}\gamma+\alpha &= 2\left(|OP|^2+r^2\right)\\ \alpha T^2+2\beta T+\gamma &= \left(T^2+1\right)|AP|^2\\ \gamma T^2-2\beta T+\gamma &= \left(T^2+1\right)|BP|^2\end{aligned}$$

so that (α,β,γ) is actually the solution of a linear system that depends on T and the squares of the distances of P to O, A and B. There is a unique solution provided that A, O and B are not aligned, i.e. $T(T^2-1)\neq 0$. This solution is:

$$\alpha=\frac{\left(|PA|^2+|PB|^2\right)\left(T^2+1\right)-4\left(|PO|^2+r^2\right)}{T^2-1},$$

and

$$\beta=\frac{\left(|PA|^2-|PB|^2\right)\left(T^2+1\right)}{T}, \quad \gamma=2\left(|PO|^2+r^2\right)-\alpha.$$

4.3.2 Integrals for convolution

Using the above parametrization of an arc of circle $\widehat{A^OB}$ the associated convolution function with the power inverse kernel $K_{2i}^{inverse}$ is

$$\mathscr{C}^{2i}_{\widehat{A^OB}}(P)=\int_{-T}^{T}\frac{1}{|P\Gamma(t)|^{2i}}\frac{2r}{1+t^2}\mathrm{d}t=2r\int_{-T}^{T}\frac{(1+t^2)^{i-1}}{(\alpha t^2-2\beta t+\gamma)^i}\mathrm{d}t=2rF_{i,0}(\alpha,\beta,\gamma,\lambda,\delta)$$

as the infinitesimal arc-length is $|\Gamma'(t)|=\frac{2r}{1+t^2}$.

We choose a radius or scale function $\rho,\Lambda:[-T,T]\to\mathbb{R}$ that is linear in the parameter t used above. A more intrinsic choice would be to have a radius or scale function linear in the arc length. Since the arc-length is $2r\arctan(t)\sim 2rt+O(t^3)$, linearity in t is a reasonable approximation for arcs defined by an angle less than π.

Convolution with varying radius according to $\rho:t\mapsto\lambda+\delta t$ is then given by

$$\mathscr{H}^{2i}_{\widehat{A^OB},\rho}(P)=\int_{-T}^{T}\frac{(\lambda+\delta t)^{2i}}{|P\Gamma(t)|^{2i}}\frac{2r}{1+t^2}\mathrm{d}t=2rF_{i,2i}(\alpha,\beta,\gamma,\lambda,\delta).$$

Convolution with scale function $\Lambda:t\mapsto\lambda+\delta t$ is given by

$$\mathscr{S}^{2i}_{\widehat{A^OB},\Lambda}(P)=\int_{-T}^{T}\frac{(\lambda+\delta t)^{2i}}{|P\Gamma(t)|^{2i}}\frac{2r}{1+t^2}\frac{\mathrm{d}t}{1+\delta t}=2rF^{-T,T}_{i,2i-1}(\alpha,\beta,\gamma,\lambda,\delta).$$

4.3.3 Closed forms through recurrence formulas

Given that

$$F_{1,0} = \left[\frac{1}{\sqrt{ca-b^2}} \arctan\left(\frac{at-b}{\sqrt{ca-b^2}} \right) \right]_{-T}^{T}$$

we can recover the expression for $F_{i,0}$, for all $i \in \mathbb{N}$, thanks to the recurrence relationship

$$2\,(i+1)\left(ac-b^2\right)F_{i+2,0} - (1+2i)\,(a+c)\,F_{i+1,0} + 2\,i\,F_{i,0} = \left[\frac{\left(1+t^2\right)^i \left(b\left(t^2-1\right)+(a-c)\,t\right)}{\left(at^2-2\,bt+c\right)^{i+1}} \right]_{-T}^{T}$$

On the other hand, the integrals $F_{i,k}$ satisfy the following recurrence:

$$a\,(k+3)\,F_{i,k+4} - A_3\,F_{i,k+3} + A_2\,F_{i,k+2} + A_1\,F_{i,k+1} + A_0\,(k+1)\,F_{i,k} = \left[\frac{\delta^3\left(1+t^2\right)^i (\lambda+\delta\,t)^{k+1}}{\left(at^2-2\,bt+c\right)^{i-1}} \right]_{-T}^{T}$$

where

$$\begin{aligned}
A_3 &= -2\,(2k+5)\,a\lambda - 2\,b\,(i+2+k)\,\delta, \\
A_2 &= 6\,(k+2)\,a\lambda^2 + 2\,b\,(5+3k+2i)\,\delta\,\lambda + ((3+k-2i)\,a + (k+2i+1)\,c)\,\delta^2, \\
A_1 &= -2\,a\,(3+2k)\,\lambda^3 - 2\,b\,(4+3k+i)\,\delta\,\lambda^2 + 2\,((i-2-k)\,a - (i+1+k)\,c)\,\delta^2\lambda + 2\,b\,(i-2-k)\,\delta^3, \\
A_0 &= \left(\delta^2+\lambda^2\right)\left(\lambda^2 a + c\delta^2 + 2\,b\lambda\,\delta\right).
\end{aligned}$$

By specializing the above equation to $k=-1$ one can obtain $F_{i,3}$ from $F_{i,2}$, $F_{i,1}$ and $F_{i,0}$. These latter are thus sufficient to determine $F_{i,k}$ for all $k \geq 4$.

To determine $F_{i,1}$ we observe that, for $i \neq 0$,

$$a\,F_{i+1,1} - F_{i,1} = (b\,\delta + a\,\lambda)\,F_{i+1,0} - \lambda\,F_{i,0} - \frac{\delta}{2\,i}\left[\frac{\left(1+t^2\right)^{\frac{i}{2}}}{\left(at^2-2\,bt+c\right)^{\frac{i}{2}}} \right]_{-T}^{T}$$

and

$$F_{1,1} = \left[\frac{(b\,\delta + a\,\lambda)}{a\sqrt{ac-b^2}} \arctan\left(\frac{at-b}{\sqrt{ac-b^2}} \right) + \frac{\delta}{2a} \ln\left(at^2-2\,bt+c\right) \right]_{-T}^{T}.$$

To determine $F_{i,2}$ we can use the linear recurrence that provides $F_{i,k+2}$ in terms of $F_{i,k+1}$, $F_{i,k}, F_{i+1,k}$, and $F_{i+2,k}$. Specialized to $k=0$ this recurrence simplifies to:

$$A_{02}\,F_{i,2} + A_{01}\,F_{i,1} + A_{00}\,F_{i,0} + A_{10}\,F_{i+1,0} + A_{20}\,F_{i+2,0} = \left[\delta\,\frac{(\lambda+\delta\,t)\left(1+t^2\right)^i}{\left(at^2-2\,bt+c\right)^{i+1}}\,C \right]_{-T}^{T}$$

where

$$
\begin{aligned}
A_{02} &= a\left((a-c)\,\delta\,\lambda + b(\delta^2-\lambda^2)\right),\\
A_{01} &= -2\,(ib\delta+\lambda\,a)\left(-\lambda^2 b+(a-c)\,\delta\,\lambda+b\delta^2\right),\\
A_{00} &= -\left(\delta^2+\lambda^2\right)\left(b\left((2i-1)\,a+2ci\right)\delta^2+\left((2i-1)\,a^2+ac+2b^2 i\right)\lambda\,\delta+\lambda^2 ab\right),\\
A_{10} &= (1+2i)\,a^2\,(a+c)\,\delta\,\lambda^3+b\left(3a^2\,(1+2i)+ac\,(3+4i)+2ib^2\right)\delta^2\lambda^2\\
&\quad+\left((4i+1)\,ca^2+a\left(c^2+2b^2\,(i+1)\right)+2b^2\,(3i+1)\,c\right)\delta^3\lambda\\
&\quad+b\left((4i+1)\,ca+(1+2i)\,c^2-2ib^2\right)\delta^4,\\
A_{20} &= -2\,\delta\,(i+1)\,(\lambda\,a+b\delta)\left(ac-b^2\right)\left(a\lambda^2+c\delta^2+2b\lambda\,\delta\right),
\end{aligned}
$$

and

$$
\begin{aligned}
C &= a^2\left(b-bt^2-(a-c)\,t\right)\lambda^2+\left(a^2bt^2-b^2\,(3a+c)\,t+b\left(c^2+2b^2\right)\right)\delta^2\\
&\quad+\left(a^2\,(a-c)\,t^2-2b\left(b^2+2a^2-ac\right)t+b^2\,(3a+c)\right)\delta\,\lambda.
\end{aligned}
$$

4.4 Conclusions

In this chapter we introduced Creative Telescoping as an approach to lower computational cost in the convolution field evaluations. We provided explicit recurrence formulas for line segments and arcs of circles, with varying radii or scale functions. The formulas have great generality and can be used for kernels of any degree. We developed formulas for power inverse kernels. Other kernel families are amenable as well. One issue with this approach is the size of the factors in the recurrence formulas which can have hundreds of terms, depending on the kernel. A numerical integration approach may be preferable for high degree kernels or when custom kernels are used.

Chapter 5

Anisotropic convolution surfaces

The content of this chapter was published in:

[Fuentes Suárez et al., 2019] Fuentes Suárez, A. J., Hubert, E., and Zanni, C. (2019). Anisotropic convoludtion surfaces. Computers & Graphics, 82:106–116.

5.1 Introduction

Skeletons, as a set of curves and/or surfaces centered inside a shape, provide a compact representation of the shape structure. Due to this property, skeletons have proved useful in many applications ranging from shape analysis to 3D modeling and deformation [Tagliasacchi et al., 2016].

Convolution surfaces [Bloomenthal and Shoemake, 1991] associate radii information to the skeleton and provide a simple way for users to rapidly define a shape. A convolution surface is an implicit surface defined as a level set of a scalar field, the convolution field, that is obtained by integrating a kernel function over the skeleton. This technique allows to build a complex shape by modeling parts that assemble into a smooth surface, independently of the smoothness of the skeleton. They also represent a volume with the convolution surface as its boundary and can therefore be combined with other composition operators from implicit modeling frameworks [Pasko et al., 1995, Wyvill et al., 1999].

Skeleton-based implicit surfaces, have been extensively used to model a variety of smooth organic shapes, such as animals and trees, either by direct skeleton manipulation [Sherstyuk, 1999a, Zhu et al., 2015a, Zanni et al., 2011] or through sketch-based modeling [Entem et al., 2015, Bernhardt et al., 2008, Zhu et al., 2011, Wither et al., 2009], or immersive modeling in virtual environments [Zhu et al., 2017].

Standard convolution surfaces, and previous extensions, allows to model a large range of shape when used with a combination of both 1D and 2D skeletons. The latter are more cumbersome to manipulate during modeling, but using only 1D skeletons is restrictive in terms of the diversity of shape that can be generated. For instance, representing muscles for 3D animation with 1D skeletons require non-circular cross sections [Roussellet et al., 2018]. Current solutions for convolution surfaces consist in adding extra skeleton pieces which adds to the complexity, or using spatial warpings, which breaks the smoothness.

We therefore introduce *anisotropic convolution surfaces*, an extension that increases the modeling freedom, providing ellipse-like normal sections around 1D skeletons. We increase the diversity of shapes that can be generated from 1D skeletons, and diminish the need for 2D skeletons, while still retaining smoothness. We achieve anisotropy not just in the normal sections but also in the tangential direction. This allows sharper and steeper radius variation, and control of thickness at skeleton endpoints.

For anisotropic convolution a frame and three radii are associated to the points on the skeletal curves. These can be defined with few parameters along any continuous curve with continuous unit tangent ($\mathscr{G}^1$ *curves*). We thus favor *circular splines*, that is $\mathscr{G}^1$ curves piecewise composed of arcs of circle or line segments, as skeletal curves. Any spatial $\mathscr{G}^1$ curve can be approximated with a circular spline [Song et al., 2009] and their Rotation Minimizing Frames [Wang and Joe, 1997] are easily computed. As compared to polylines, circular spline require less pieces to obtain a good approximation of skeletal curves with high curvature or torsion, or to obtain visual smoothness of the resulting

convolution surface. This furthermore reduces the number of integrals to evaluate for the convolution field. It nonetheless retains a fast computation of the distance from a point to the curve.

We demonstrate the advantages of anisotropic convolutions in three applications, skeleton-based modeling, general implicit modeling, and shape approximation. For the first application we introduce a meshing technique based on a *scaffolding* method [Fuentes Suárez and Hubert, 2018b]. A *scaffold* is a coarse quad mesh that is built around a skeleton made of line segments. Here we extend the scaffolding method from line segments to $\mathscr{G}^1$ curves. We construct the mesh by first computing a refined scaffold that is then projected onto the surface. The projection step is done by ray shooting from the skeleton. This meshing technique can provide a coarse to fine quad mesh, that matches the topology of the skeleton and has good edge-flow. We aim to keep the number of evaluations of the convolution field reasonably low, at least comparable to Marching Cubes [Lorensen and Cline, 1987] and variants [Gomes et al., 2009, Wenger, 2013].

Other than the simple and intuitive modeling of shapes, we also investigate the use of anisotropic convolution surfaces for shape approximation. Starting from a mesh of the shape and its skeletonization [Livesu and Scateni, 2013, Yan et al., 2016], we propose a pipeline to best fit an anisotropic convolution surface. The new shape so obtained has a compact representation, and can be seen as a lossy compression of the original model. It is also smooth and can be obtained from occluded models (with holes). We use circular splines approximations to simplify the output of skeletonization algorithms (which usually requires a hefty post-processing [Barbieri et al., 2016]), and call on optimization to determine the parameters of the anisotropic convolution surface.

5.1.1 Related work

Several methods have been proposed for the generation of surfaces around 1D skeletons: sweep surfaces [Requicha, 1980], offset surfaces [Pham, 1992], canal surfaces [Peternell and Pottmann, 1997, Fryazinov and Pasko, 2017], B-meshes [Ji et al., 2010], and convolution surfaces [Blinn, 1982, Zanni, 2013] are some of them.

Convolution surfaces can be defined around any piecewise regular curve. To lower the complexity of the formulas of the integrals forming the convolution field one resorts to simpler curves, as line segments and arcs of circle [Fuentes Suárez and Hubert, 2018a, Hornus et al., 2003, Hubert and Cani, 2012, Jin et al., 2001, Jin and Tai, 2002a, McCormack and Sherstyuk, 1998, Sherstyuk, 1999b, Zanni, 2013, Zanni et al., 2013, Zhu et al., 2011], or even quadratic curves [Jin and Tai, 2002b]. More complex skeletal curves are approximated or warped [Zanni, 2013]. Though line segments and arcs of circle support closed form formulas for convolution fields [Hubert and Cani, 2012, Hubert, 2012, Jin and Tai, 2002b, Jin and Tai, 2002a, Sherstyuk, 1999b, Zanni, 2013], they can be daunting when varying the radius in a scale

invariant way [Fuentes Suárez and Hubert, 2018a]. Further efforts in that direction are unproductive for the anisotropic formulation we present. Thus we resort to numerical integration [Piessens et al., 1983], and this further motivates the development of a meshing technique requiring less evaluations of the the convolution field.

Weighted convolution and alternative formulations were introduced [Jin and Tai, 2002a, Hornus et al., 2003, Zanni et al., 2013] to controllably vary the thickness along the skeleton, still with close-to-circular normal sections. Tai *et al.* [Tai et al., 2004] introduced general shaped normal sections with a modeling technique based on field remapping, at the cost of smoothness and complexity. Another approach for anisotropy is B-Meshes [Ji et al., 2010], where ellipse-like normal sections are achieved by interpolating ellipsoids along line segments. Nonetheless the orientation of the ellipsoid seems rigidly imposed by the system. The smoothness is handled on a post-processing step.

In [Jin and Tai, 2002b] *planar* circular splines were first used for convolution surface skeletons. Our *spatial* circular spline approach takes advantage of biarcs theory [Bolton, 1975]. We adapt some ideas on Rotation Minimizing Frames [Bolton, 1975] for sweep surfaces [Wang and Joe, 1997] to the context of convolution surfaces. Biarcs-based circular spline ideas are also discussed in [Meek and Walton, 2008, Song et al., 2009]. To model organic shapes with few primitives [Zanni et al., 2011] introduced helical primitives with warping, while our suggestion is to use approximation of helices with circular splines, removing the need for warping.

Motivated by the limitations of 1D skeleton, convolution surfaces around 2D skeletons have been developed [Jin et al., 2008, Hubert, 2012, Zhu et al., 2011, Sherstyuk, 1999a] for shape approximation and other applications. They are inherently more difficult to model and present more complex formulas. Our approach can be extended to 2D skeletons. Nonetheless the modeling capabilities of *anisotropic convolution* diminish the need of 2D skeletons.

Different scaffold constructions have been used in the literature [Angelidis and Cani, 2002, Bærentzen et al., 2012, Fuentes Suárez and Hubert, 2018b, Ji et al., 2010, Panotopoulou et al., 2018, Usai et al., 2015, Yao et al., 2009]. The algorithm in [Fuentes Suárez and Hubert, 2018b] works for line segments skeletons of any topology and can respect their symmetries. We provide an adaptation of [Fuentes Suárez and Hubert, 2018b] for $\mathscr{G}^1$ curves.

5.1.2 General overview and contributions

Anisotropic convolution uses *frames* on the skeletal curves, which thus need to be $\mathscr{G}^1$ curves. We start by discussing *frames* for $\mathscr{G}^1$ curves, biarc theory, and circular splines in Section 5.2. An introduction to convolution surfaces, extensions, and our new formulation are discussed in Section 5.3. The meshing process that makes use of scaffolding is

discussed in Section 5.4. In Section 5.5 we describe the numerical evaluation of the convolution fields. Section 5.6 describes three applications: skeleton-based modeling, general implicit modeling, and shape approximation.

Our contributions are: an extension to the modeling capabilities of convolution surfaces techniques; the introduction of a spatial circular spline skeletal description that allows for the definition of shapes in an organic way; the description of a meshing technique that uses the skeletal information coupled with a scaffold to improve the polygonization of the surfaces; and applications of anisotropic convolution to the modeling and approximation of a greater variety of shapes.

Notation Vectors are denoted by small bold letters and are assumed to be in column form. $\|\cdot\|$ denotes the Euclidean norm. Matrices are 3×3 and represented by capital bold letters. Scalars are represented by non-bold letters. $\mathbf{I}$ denotes the identity matrix. $\mathbf{A}^i$ refers to the i-th column vector of the matrix $\mathbf{A}$. $\mathrm{SO}(3)$ and Sym^+, denote the group of orthonormal matrices, and the cone of positive-definite symmetric matrices, respectively. $\mathrm{Diag}(a,b,c)$ denotes the diagonal matrix with a,b,c in the main diagonal. A piecewise-regular curve Γ is said to be $\mathscr{G}^1$ if it has continuous unit tangent. We assume all curves to be parametrized by arc-length.

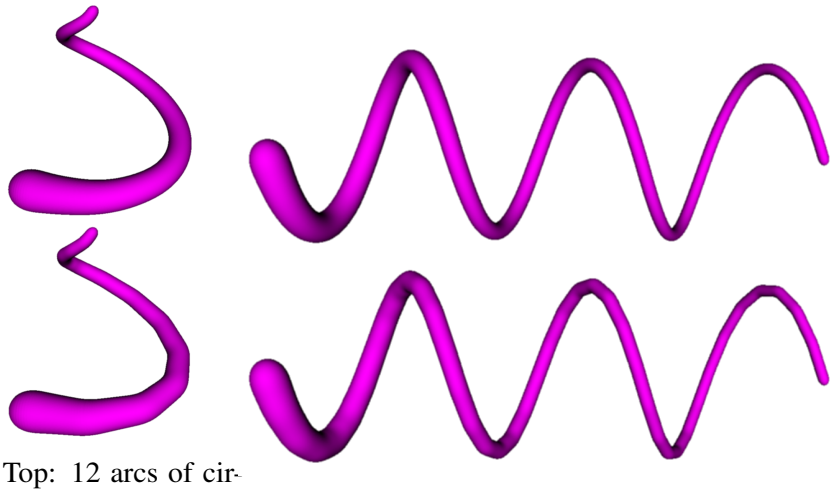
5.2 Preliminaries: circular splines and frames

In this section we discuss preliminary concepts that are going to be used later in both theoretical and practical developments. Following [Wang and Joe, 1997], we briefly present the main components of *circular splines* approximation, and key properties of *frames* in the context of convolution.

5.2.1 Circular splines: approximation of general curves

Line segments may require many pieces to obtain a good approximation and a visually appealing convolution surface. Furthermore, a polyline approximation of a curve is continuous *at most*. Using arcs of circle and line segments one can get a $\mathscr{G}^1$ approximation of a skeletal curve in the form of circular splines [Wang and Joe, 1997]. Figure 5.1 compares the convolution of polylines and circular splines. We use biarc theory to construct a circular spline approximation from *Hermite data*.

Let $\mathscr{H} = (A_0, \mathbf{t}_0, A_1, \mathbf{t}_1)$ be the *Hermite data* defined by a pair of points $A_0, A_1 \in \mathbb{R}^3$, $A_0 \neq A_1$, along with their associated unit tangent vectors $\mathbf{t}_0, \mathbf{t}_1$. An *interpolating biarc* [Bolton, 1975, Meek and Walton, 2008, Wang and Joe, 1997] of $\mathscr{H}$ is then defined as a $\mathscr{G}^1$ curve joining A_0 and A_1 formed by two arcs of circle such that one arc is passing through A_0 with unit tangent $\mathbf{t}_0$, and the other is passing through A_1 with unit tangent $\mathbf{t}_1$ (see Figure 5.2). Biarcs are simple but powerful enough for modeling spatial



(a) Top: 12 arcs of circle; bottom: 14 line segments.

(b) Top: 42 arcs of circle; bottom: 42 line segments.

Figure 5.1: Convolution surfaces around $\mathscr{G}^1$ circular spline (top), and polyline (bottom), approximations of (a) the spiral $(\frac{1}{2}t\cos t, \frac{3}{4}t\sin t, \frac{4}{5}t)$, $t \in [0, 2\pi]$, and (b) the elliptical helix $(2\cos t, 3\sin t, t)$, $t \in [0, 6\pi]$. All surfaces are smooth but the surfaces around a circular spline approximation are "visually" smoother.

curves [Song et al., 2009]. They allow interpolation of a set of Hermite data with a $\mathscr{G}^1$ curve, have an analytical formula for arc-length, and the distances from points in space are easy to compute [Song et al., 2009].

On each Hermite data, the fitting biarcs form a one-parameter family [Song et al., 2009]. There are several criteria [Nutbourne and Martin, 1988, Chapter 3] to choose a particular biarc. In this work we use the biarc that gives equal tangent length for its two arcs ($l_0 = l_1$ in Figure 5.2). Our choice follows [Song et al., 2009] where a circular spline approximation is computed from sampling points. We reuse the equal tangent property when computing a circular spline approximation of a polyline in Section 5.6 (Figure 5.22b).

Given the Hermite data set $\{\mathscr{H}_i = (A_i, \mathbf{t}_i, A_{i+1}, \mathbf{t}_{i+1})\}_{i=0}^{n-1}$, we can now construct an interpolating circular spline [Song et al., 2009]. For each $\mathscr{H}_i$ we take the corresponding biarc, except when $\mathbf{t}_0 = \mathbf{t}_1 = A_1 - A_0$, where a line segment is used instead. The $\mathscr{G}^1$ curve piecewise defined by the biarcs and line segments is the interpolating *circular spline*. For a general $\mathscr{G}^1$ curve, a sampling is used to obtain the Hermite data set. The number of samples can be increased to get a more accurate approximation [Song et al., 2009] (Figure 5.3). In the rest of the paper we assume the skeletal curves to be circular splines.

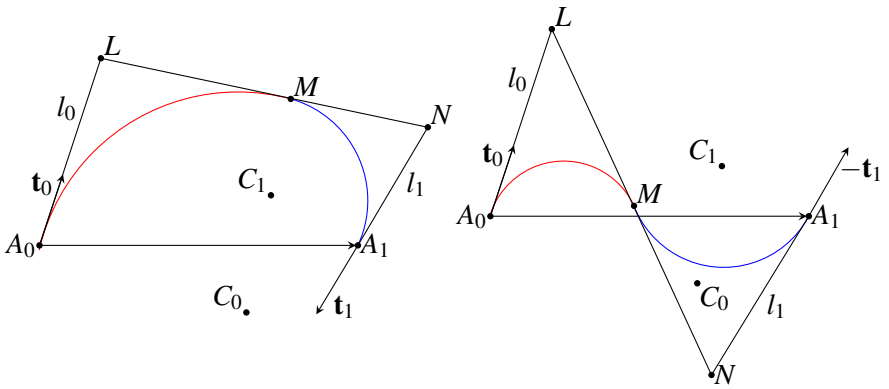


Figure 5.2: The biarc construction for $\mathscr{H} = (A_0, \mathbf{t}_0, A_1, \mathbf{t}_1)$, on the left; and $\mathscr{H} = (A_0, \mathbf{t}_0, A_1, -\mathbf{t}_1)$, on the right. Notice that each biarc has a natural tangential polyline: A_0LMNA_1.

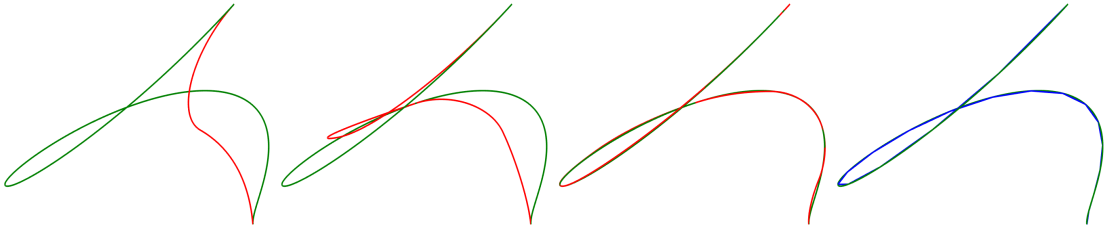


Figure 5.3: Biarcs interpolation (red) of a curve (green) with (from left to right) 2, 3, and 5 sampling points (2, 4, 8 arcs). The last picture on the right shows a polyline interpolation with 20 sampling points.

5.2.2 Frames for $\mathscr{G}^1$ curves

A *frame* is an orthonormal set of three vectors defined along a regular curve such that the first one coincides with the unit tangent. Formally, for a regular curve $\Gamma : [0,l] \to \mathbb{R}^3$, a *frame* is a continuous map $\mathbf{F} : [0,l] \to \mathrm{SO}(3)$ such that:

$$\mathbf{F}^1(s) = \Gamma'(s)^\mathsf{T} \quad \forall s \in [0,l]. \tag{5.2.1}$$

Framing a curve is a well-studied problem [Bishop, 1975, Wang and Joe, 1997] with the most common frame being the *Frenet* frame for $\mathscr{G}^2$ curves. Of particular interest for us are the *Rotation Minimizing Frames* (RMF) that can be defined for $\mathscr{G}^1$ curves and minimize the rotation around the tangent direction.

When Γ is a $\mathscr{G}^1$ curve, piecewise composed of regular curves $\Gamma_h : [0,l_h] \to \mathbb{R}^3$ ($h = 1,2,\ldots,d$), we can define a frame $\mathbf{F}$ for Γ by means of the frames $\mathbf{F}_h$ of Γ_h. Let $L_h = l_1 + l_2 + \cdots + l_h$ and $L_0 = 0$, then $\Gamma : [0,L_d] \to \mathbb{R}^3$ is parameterized as

$$\Gamma(s) = \Big\{ \Gamma_h(s) \quad L_{h-1} \le s < L_h \quad (h = 1,2,\ldots,d)$$

and $\mathbf{F}$ is defined as

$$\mathbf{F}(s) = \left\{ \tilde{\mathbf{F}}_h(s - L_{h-1}) \quad L_{h-1} \le s < L_h \quad (h = 1, 2, \ldots, d) \right. \tag{5.2.2}$$

where $\tilde{\mathbf{F}}_h(s) = \mathbf{F}_h(s)\mathbf{R}_h$, with $\mathbf{R}_0 = \mathbf{I}$ and $\mathbf{R}_h \in \mathrm{SO}(3)$ is the rotation matrix that takes $\mathbf{F}_h(0)$ to $\tilde{\mathbf{F}}_{h-1}(l_{h-1})$ by rotating around $\mathbf{F}^1_h(0) = \mathbf{F}^1_{h-1}(l_{h-1})$, that is $\tilde{\mathbf{F}}_{h-1}(l_{h-1}) = \mathbf{F}_h(0)\mathbf{R}_h$ (see Figure 5.4).

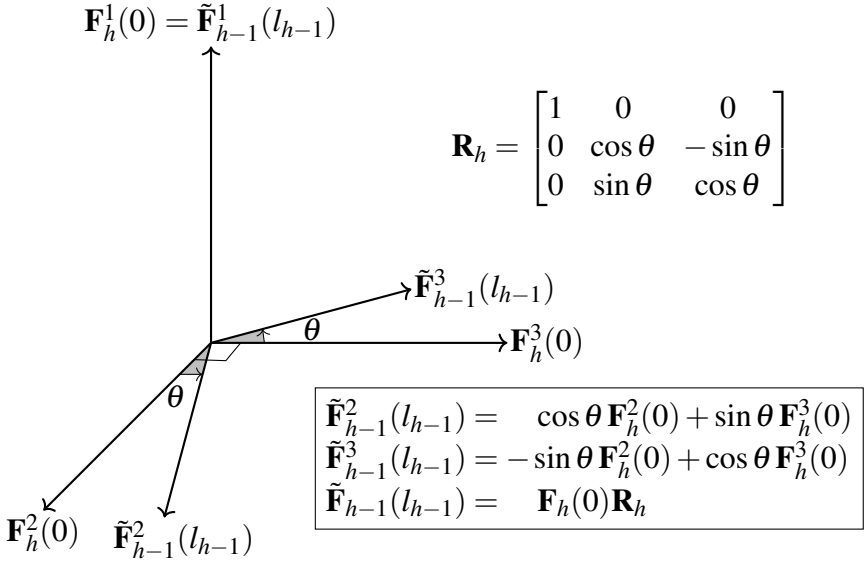


Figure 5.4: Rotation of $\mathbf{F}_h(0)$ into $\tilde{\mathbf{F}}_{h-1}(l_{h-1})$. Since Γ is a $\mathscr{G}^1$ curve, the unit tangent vectors at $\Gamma_h(l_h)$ and $\Gamma_{h+1}(0)$ coincide, thus $\mathbf{F}^1_h(l_h) = \mathbf{F}^1_{h+1}(0)$ $(h = 1, 2, \ldots, d-1)$, and $\mathbf{F}_{h+1}(0)$ differs from $\mathbf{F}_h(l_h)$ by a rotation R_h around the axis $\mathbf{F}^1_{h+1}(0) = \mathbf{F}^1_h(l_h)$. The rotation is needed in order to keep a continuous frame along Γ.

When $\mathbf{F}_h$ is a RMF of Γ_h, the frame defined in (5.2.2) is a RMF for the whole curve Γ [Bishop, 1975, Wang and Joe, 1997]. For arcs of circles, the Frenet frame is a RMF. It consists of the unit tangent, a unit radial vector and a normal vector to the plane where the arc sits. Note that a RMF may not be continuous for closed curves. A corrective rotation is to be added along the curve to obtain a continuous frame [Wang and Joe, 1997]. We later show how this is handled in a natural way in anisotropic convolution ($\theta_1 \neq \theta_0$ in Equation (5.3.9)).

5.3 Convolution surfaces: from varying thickness to anisotropy

Generally speaking a *convolution surface* in $\mathbb{R}^3$ is the level set of a *convolution* field that results from the integration of a *kernel* function K along a *skeleton* Σ. A *kernel* is a

function $K : \mathbb{R}^+ \to \mathbb{R}^+$ which is at least $\mathscr{C}^1$, and is strictly decreasing when nonzero.

In this paper the skeletons are collections of regular curves with finite arc-length that intersect only at endpoints. For a skeletal curve $\Gamma : [0,l] \to \mathbb{R}^3$, the *convolution field* C^K_Γ is defined as

$$C^K_\Gamma(P) = \int_0^l K\left(\|P - \Gamma(s)\|\right) \mathrm{d}s \quad \text{for all } P \in \mathbb{R}^3. \tag{5.3.1}$$

For the skeleton Σ, the *convolution field* C_Σ is defined as

$$C_\Sigma = \sum_{\Gamma \in \Sigma} \delta_\Gamma C^K_\Gamma(P), \tag{5.3.2}$$

with $\delta_\Gamma \in \mathbb{R}$ for all $\Gamma \in \Sigma$. In the simplest case $\delta_\Gamma = 1$ for all $\Gamma \in \Sigma$. When $\delta_\Gamma < 0$ a *soft carving* effect is achieved (like in [Tai et al., 2004], see Figure 5.10c for an example). Equation (5.3.1) can be written as a line integral, therefore the convolution field (5.3.2) is independent of the parameterization and skeleton subdivisions. Leibniz integral rule for differentiation under the integral symbol [Flanders, 1973] guarantees that C^K_Γ is also as smooth as K for points outside of the skeleton (i.e. for $P \notin \Gamma([0,l])$).

The *convolution surface* $\mathscr{S}^c_\Sigma$, for the level value $c > 0$, is formally defined as

$$\mathscr{S}^c_\Sigma = \left\{ P \in \mathbb{R}^3 \;\middle|\; \mathscr{C}^K_\Sigma(P) = c \right\}. \tag{5.3.3}$$

It is closed (in a topological sense) and smooth, provided c is not a critical value of C^K_Σ [do Carmo, 1976, Section 2-2]. For simplicity if Σ has only one skeletal curve Γ we identify Σ with Γ.

Among the kernel functions proposed in the literature [Hubert, 2012, Hubert and Cani, 2012, Sherstyuk, 1999b, Sherstyuk, 1999a, Zanni et al., 2013, Wyvill et al., 1986, Zanni, 2013] , we choose the compactly supported polynomial kernel:

$$K(x) = \begin{cases} \frac{35}{16}\left(1 - x^2\right)^3 & 0 \le x \le 1 \\ 0 & \text{otherwise.} \end{cases} \tag{5.3.4}$$

The integral of the kernel over its support is 1, that is $\int_0^1 K(x)\mathrm{d}x = 1$. Compactly supported polynomial kernels are advantageous because changes on the skeleton affect only locally the convolution surface while the kernel is still a simple function (piecewise polynomial). A similar type of kernel was recently used in the Brush2Model tool [Zhu et al., 2017].

5.3.1 Varying thickness

Despite the freedom in the choice of the kernel and level set, the thickness around the skeleton in a convolution surface do not change very much along the supporting skeletal curve. A convolution surface around a line segment has a tubular structure with circular normal sections.

In order to control the thickness several alternatives have been introduced. One of the first ideas was to use a *weight* function. Polynomial weight functions were used in [Hubert and Cani, 2012, Jin and Tai, 2002a, Jin and Tai, 2002b, Jin et al., 2001]. Other alternatives, proposed by Hornus *et al.* [Hornus et al., 2003], and SCALIS proposed by Zanni *et al.* [Zanni et al., 2013], modify the distance from the point to the curve before evaluating the kernel, the difference is in the normalization factor introduced in SCALIS. A summary of the convolution surface variants is shown in Table 5.1. All variants have circular normal sections. Within this limitation, SCALIS offers a better blending behavior and control over small details on which we build.

Variant	**Formulation**
WEIGHTED	$J^K_{\Gamma,w}(P) = \int_0^l w(s)K(\|P-\Gamma(s)\|)\,\mathrm{d}s$
[Jin et al., 2001, Hubert and Cani, 2012]	$w:[0,l]\to\mathbb{R}$ *weight function*
HORNUS	$H^K_{\Gamma,\rho}(P) = \int_0^l K\left(\frac{\|P-\Gamma(s)\|}{\rho(s)}\right)\mathrm{d}s$
[Hornus et al., 2003]	$\rho:[0,l]\to\mathbb{R}^+$ *radius function*
SCALIS	$Z^K_{\Gamma,\lambda}(P) = \int_0^l K\left(\frac{\|P-\Gamma(s)\|}{\lambda(s)}\right)\frac{\mathrm{d}s}{\lambda(s)}$
[Zanni et al., 2013]	$\lambda:[0,l]\to\mathbb{R}^+$ *scale function*

Table 5.1: Convolution surface variants. Each alternative formulation uses a function that controllably varies the thickness along the skeleton.

5.3.2 Anisotropic convolution

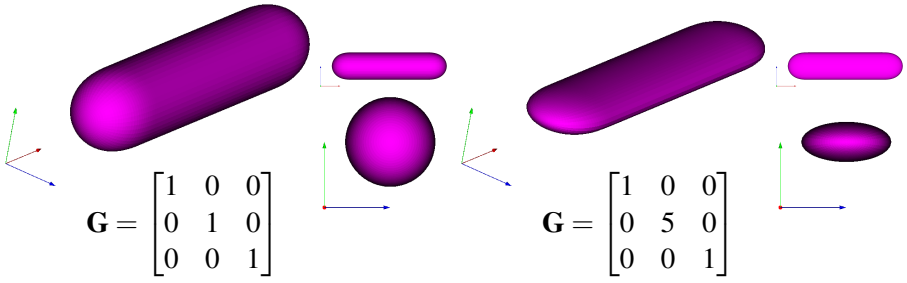
In order to increase modeling freedom we introduce anisotropy in the convolution surfaces. The idea is to change the way the distance to a point in the skeleton is computed before the kernel evaluation. Instead of using the standard Euclidean (isotropic) distance, we introduce an anisotropic distance. This is achieved with a scalar product defined by a *metric matrix*, related to a frame of the skeletal curve.

A matrix $\mathbf{G}\in Sym^+$ defines a scalar product $\langle\cdot,\cdot\rangle$ in $\mathbb{R}^3$ as $\langle\mathbf{x},\mathbf{y}\rangle = \mathbf{x}^\mathsf{T}\mathbf{G}\mathbf{y}$ for all $\mathbf{x},\mathbf{y}\in\mathbb{R}^3$. Thus $\mathbf{x}\mapsto\sqrt{\mathbf{x}^\mathsf{T}\mathbf{G}\mathbf{x}}$ defines a norm (and hence a distance) in $\mathbb{R}^3$. We refer to $\mathbf{G}$ as a *metric matrix*. The *anisotropic convolution field* is then defined as

$$C^K_{\Gamma,\mathbf{G}}(P) = \int_0^l K\circ g(\Gamma(s),P-\Gamma(s))\cdot g(\Gamma(s),\Gamma'(s))\mathrm{d}s, \tag{5.3.5}$$

where $g(\mathbf{y},\mathbf{x}) = \sqrt{\mathbf{x}^\mathsf{T}\cdot\mathbf{G}(\mathbf{y})\cdot\mathbf{x}}$ for all $(\mathbf{y},\mathbf{x})\in\Gamma([0,l])\times\mathbb{R}^3$, and $\mathbf{G}:\Sigma\to Sym^+$ is a map that assigns a metric matrix to each point in the skeleton. In practice $\mathbf{G}$ is seen as a

map from $[0,l]$ to Sym^+. If $\mathbf{G}(s)=\mathbf{I}$, then (5.3.5) reduces to (5.3.1). If $\mathbf{G}(s)=\lambda(s)^{-2}\mathbf{I}$, then (5.3.5) reduces to SCALIS formulation (Table 5.1). Other choices change the anisotropy of the surface. The normal sections are ellipses, and we can, for example, obtain a flattened volume (Figure 5.5). To vary the thickness we vary the metric matrix along the skeletal curve Γ.



(a) Isotropic metric. (b) Anisotropic metric.

Figure 5.5: Constant metric matrices along a line segment.

Anisotropic convolution has the same *scale invariant* behavior of SCALIS [Zanni et al., 2013], that we state in the following proposition (proof in supplementary material).

Proposition 5.3.1 (Scale invariance)**.** *For any scale* $\tau>0$, $C^K_{\Gamma,\mathbf{G}}(X)=C^K_{\tau\Gamma,\tau\mathbf{G}}(\tau X)$ *for all* $X\in\mathbb{R}^3$, *with* $(\tau\Gamma)(s)=\tau\Gamma(\frac{s}{\tau})$ *and* $(\tau\mathbf{G})(s)=\frac{1}{\tau^2}\mathbf{G}(\frac{s}{\tau})$ *for* $s\in[0,\tau l]$.

Proof.

$$
\begin{aligned}
C^K_{\tau\Gamma,\tau\mathbf{G}}(\tau X) &= \int_0^{\tau l} K\left(\sqrt{(\tau X-(\tau\Gamma)(s))^{\mathsf{T}}\cdot(\tau\mathbf{G})(s)\cdot(\tau X-(\tau\Gamma)(s))}\right)\cdot \\
&\quad\cdot\sqrt{(\tau\Gamma)'(s)^{\mathsf{T}}\cdot(\tau\mathbf{G})(s)\cdot(\tau\Gamma)'(s)}\,\mathrm{d}s \\
&= \int_0^{\tau l} K\left(\sqrt{\tau\left(X-\Gamma\left(\frac{s}{\tau}\right)\right)^{\mathsf{T}}\cdot\frac{1}{\tau^2}\mathbf{G}\left(\frac{s}{\tau}\right)\cdot\tau\left(X-\Gamma\left(\frac{s}{\tau}\right)\right)}\right)\cdot \\
&\quad\cdot\sqrt{\Gamma'\left(\frac{s}{\tau}\right)^{\mathsf{T}}\cdot\frac{1}{\tau^2}\mathbf{G}\left(\frac{s}{\tau}\right)\cdot\Gamma'\left(\frac{s}{\tau}\right)}\,\mathrm{d}s \\
&= \int_0^{l} K\left(\sqrt{(X-\Gamma(u))^{\mathsf{T}}\cdot\mathbf{G}(s)\cdot(X-\Gamma(u))}\right)\sqrt{\Gamma'(u)^{\mathsf{T}}\cdot\frac{1}{\tau^2}\mathbf{G}(u)\cdot\Gamma'(u)}\,\tau\,\mathrm{d}u \\
&\left[\text{taking } u=\frac{s}{\tau}\right] \\
&= \int_0^{l} K\left(\sqrt{(X-\Gamma(u))^{\mathsf{T}}\cdot\mathbf{G}(s)\cdot(X-\Gamma(u))}\right)\sqrt{\Gamma'(u)^{\mathsf{T}}\cdot\mathbf{G}(u)\cdot\Gamma'(u)}\,\mathrm{d}u \\
&= C^K_{\Gamma,\mathbf{G}}(X)
\end{aligned}
$$

□

We exploit the eigen-decomposition [Axler, 2015] of $\mathbf{G}$ to achieve control over the shape along Γ. Any matrix $\mathbf{G} \in Sym^+$ can be written as $\mathbf{G} = \mathbf{U}\mathbf{D}\mathbf{U}^\mathsf{T}$ where $\mathbf{U} \in \mathrm{SO}(3)$ has the eigen-vectors of $\mathbf{G}$ as columns, and $\mathbf{D}$ is a diagonal matrix with the eigen-values of $\mathbf{G}$ in the main diagonal. A visual representation is shown in Figure 5.6.

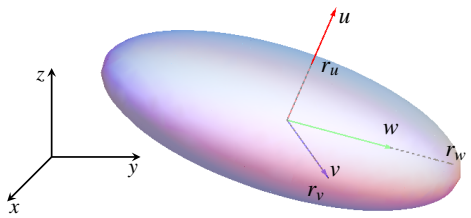


Figure 5.6: $\mathbf{G} = \mathbf{U}\mathbf{D}\mathbf{U}^\mathsf{T}$ as the ellipsoid $\mathbf{x}^\mathsf{T}\mathbf{G}\mathbf{x} = 1$. The eigenvectors $\mathbf{u}, \mathbf{v}, \mathbf{w}$ define the axes, the radii are given by the eigenvalues: $r_u = \alpha^{-\frac{1}{2}}, r_v = \beta^{-\frac{1}{2}}, r_w = \gamma^{-\frac{1}{2}}$.

Starting with a set of three orthonormal vector fields $\mathbf{u}, \mathbf{v}, \mathbf{w} : [0,l] \to \mathbb{R}^3$ and three positive functions $\alpha, \beta, \gamma : [0,l] \to \mathbb{R}^*_+$ we can define a metric matrix

$$\begin{aligned} &\mathbf{G}(s) = \mathbf{U}(s)\mathbf{D}(s)\left(\mathbf{U}(s)^\mathsf{T}\right) \quad \text{for } s \in [0,l], \text{ with} \\ &\mathbf{U}(s) = [\mathbf{u}(s)\ \mathbf{v}(s)\ \mathbf{w}(s)] \text{ and} \\ &\mathbf{D}(s) = \mathrm{Diag}(\alpha(s), \beta(s), \gamma(s)). \end{aligned} \tag{5.3.6}$$

Equation (5.3.6) decouples the magnitude ($\mathbf{D}$) and orientation ($\mathbf{U}$) of $\mathbf{G}$. Given a desired final shape, we construct a suitable map $\mathbf{G} : [0,l] \to Sym^+$ that controls the shape along Γ. To achieve this we take $\mathbf{U}$ to be a frame of Γ (Section 5.2.2) and so we get a metric matrix that "follows" the skeletal curve. With this choice, $\mathbf{U}^1(s) = \Gamma'(s)$, and we get that (5.3.5) simplifies to

$$C^K_{\Gamma,\mathbf{G}}(P) = \int_0^l K \circ g(\Gamma(s), P - \Gamma(s)) \sqrt{\alpha(s)} \mathrm{d}s. \tag{5.3.7}$$

The right hand side of (5.3.7) can be written as the line integral $\int_\Gamma K \circ g(\mathbf{y}, P - \mathbf{y}) \sqrt{\alpha(\mathbf{y})} \mathrm{d}\mathbf{y}$, and we can deduce the same advantages as in (5.3.1). We discuss next how to define $\mathbf{U}(s)$ and $\mathbf{D}(s)$ given some design parameters.

Modeling with anisotropic convolution

Anisotropic convolution allows for an intuitive modeling framework. First, to effectively describe the shape along the skeletal curve $\Gamma([0,l])$, we restrict $\mathbf{U}$ to be a frame of Γ. In practice $\mathbf{U}$ is defined as a rotation of an automatically computed frame $\mathbf{F}$ of Γ (a RMF, or Frenet frame). At each extremity of Γ the user choses the rotation and radii. Then the

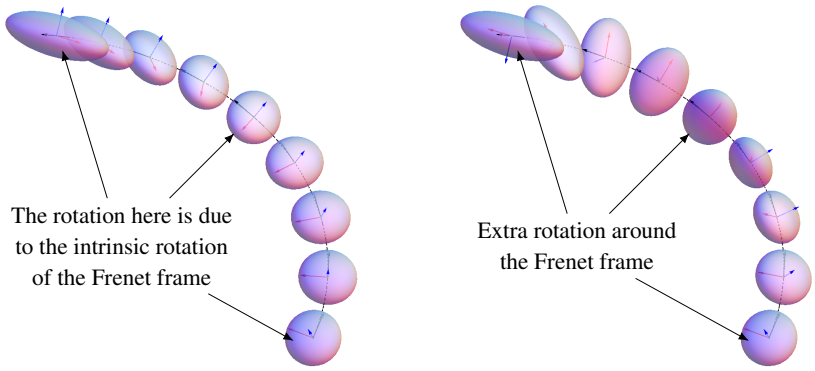
parameters (the rotation and radii) defining the matrix G are linearly interpolated along Γ. We model the surfaces as to not intersect the skeleton. For long enough skeletal curves, we have $C^K_{\Gamma,\mathbf{G}}(P) \in [1,2]$ for all points P in the skeleton, i.e. $P \in \Gamma([0,l])$. It follows that the level value c must be in the interval $(0,1)$ such that the whole skeleton is contained inside the surface.

We denote $\theta_i \in \mathbb{R}$ $(i = 0,1)$ the angles associated to the initial $(i = 0)$ and final $(i = 1)$ endpoints of Γ respectively. Then the frame $\mathbf{U}$ is defined as

$$\mathbf{U}(s) = \mathbf{F}(s)\mathbf{R}(s), \text{ with} \tag{5.3.8}$$

$$\mathbf{R}(s) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta(s) & -\sin\theta(s) \\ 0 & \sin\theta(s) & \cos\theta(s) \end{bmatrix}, \ \theta(s) = \frac{l-s}{l}\theta_0 + \frac{s}{l}\theta_1. \tag{5.3.9}$$

From user inputs, we determine initial and final eigen-values $\alpha_i, \beta_i, \gamma_i \in R^+ (i = 0,1)$ which are interpolated in order to define the positive eigen-values of $D(s)$ along the skeleton: $(\alpha(s), \beta(s), \gamma(s))$. We discuss this in Section 5.3.2.



(a) Anisotropy: $\alpha_0 = \alpha_1 = 1, \beta_0 = 1, \beta_1 = \frac{1}{4}, \gamma_0 = \gamma_1 = 2$. No rotation.

(b) Anisotropy: $\alpha_0 = \alpha_1 = 1$, $\beta_0 = 1, \beta_1 = \frac{1}{4}$, $\gamma_0 = \gamma_1 = 2$, $\theta_0 = 0, \theta_1 = \pi$.

Figure 5.7: Interpolation of the metric matrix $\mathbf{G}$ along the helix $\Gamma(t) = (5\cos t, 5\sin t, 3t), t \in [0,\pi]$ with respect to the Frenet frame of Γ.

The eight parameters $(\theta_i, \alpha_i, \beta_i, \gamma_i)$ $(i = 0,1)$ completely define $\mathbf{G} : [0,l] \to Sym^+$, and control the shape along the skeletal curve. Figure 5.7 illustrates the evolution of the parameters in (5.3.8) and (5.3.10) along the skeleton. β and γ describe the shape in the two normal directions of the frame along the curve, while α controls the distance from the tips of the surface to the extremities of the skeleton. Twisting (θ) and varying thickness (β, γ) can be combined as shown in Figure 5.8. Varying α can be used to control the "bumping" in the blending area between two pieces, as shown in Figure 5.9.

When $\mathbf{F}$ is a RMF of Γ, the frame $\mathbf{U}$ in (5.3.6) rotates minimally around Γ outside the linearly prescribed rotation given by θ_0 and θ_1. This reflects the non-intrinsic user-defined rotation of the metric matrix so as to model some twisting (figures 5.8, 5.10b, 5.17

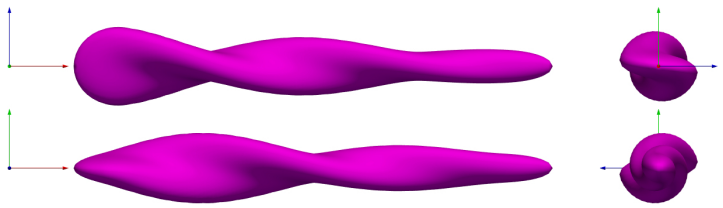
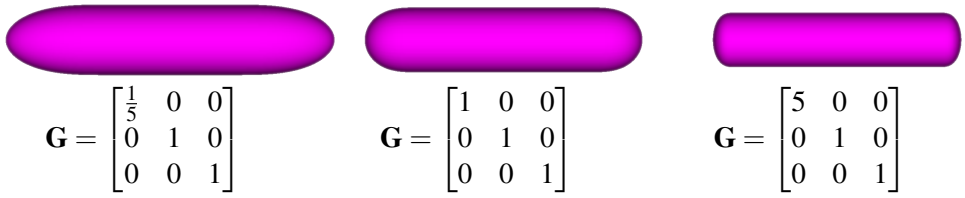


Figure 5.8: Anisotropic convolution around a line segment with a combination of twisting and varying thickness: $\alpha_0 = \alpha_1 = 1, \beta_0 = \beta_1 = 10, \gamma_0 = 1, \gamma_1 = 6, \theta_0 = 0, \theta_1 = 2\pi$.

and 5.20), or to adjust a frame for continuity along a closed curve [Wang and Joe, 1997, Wang et al., 2008] (Figure 5.19).



$$\mathbf{G} = \begin{bmatrix} \frac{1}{5} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(a) Increased.

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(b) Standard.

$$\mathbf{G} = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(c) Reduced.

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(d) "Bumping" blending.

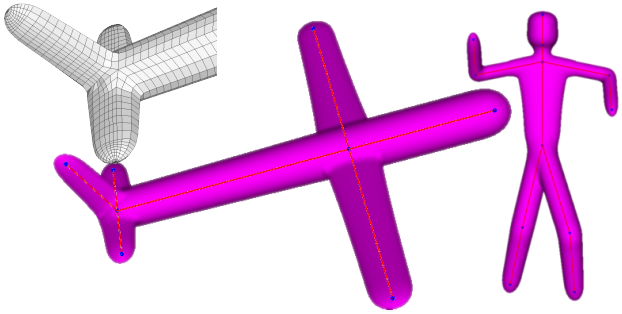
$$\mathbf{G} = \begin{bmatrix} 8 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(e) "Smoother" blending.

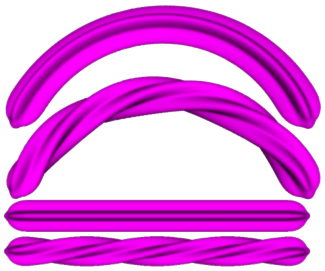
Figure 5.9: Anisotropic convolution can be used to change the extremities (tips) of the surfaces (top row). An application to reducing "bumping" in the blending area is shown in (d-e).

With anisotropic convolution one can mimic the effects of 2D skeletons. Figure 5.10a shows some shapes previously modeled in [Zhu et al., 2011] with 2D polygonal skeletons. We can also obtain a variety of centrally symmetric normal sections by reusing the same skeleton with several metrics. Figure 5.10b shows cross-shaped normal sections. In Figure 5.10c we illustrate soft carving.

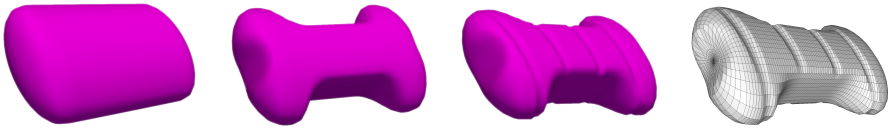
Since anisotropic convolution is scale-invariant (Proposition 5.3.1) we inherit the advantages of SCALIS [Zanni et al., 2013] such as modeling at different scales. Compared to SCALIS we gain anisotropy and twisting behavior. We can also control the radius in the tangent direction, hence the distance from the surface to the extremities of the skeletal curve; this was done before by modifying the skeletal curve [Zanni et al., 2013]. Radii control for anisotropic convolution is discussed in the next section.



(a) Simulating 2D skeletons.



(b) Cross-shaped normal sections.



(c) Soft carving effect by subtracting anisotropic fields.

Figure 5.10: Extended modeling capabilities of anisotropic convolution.

Controlling the radii

The eigenvalues and rotation of the metric matrix define the general shape of the surface. To gain a finer control over the actual radii we analyze what happens for a particular level set in the ideal case of a line segment skeleton with a constant metric matrix. We discuss next how the eigenvalue parameters must be chosen in order to attain a desired radius.

Let $\Gamma(s) = Q + s\,\mathbf{u}$ be a line segment skeleton, with $s \in [0,l]$, $\mathbf{u} \in \mathbb{R}^3$ the unit tangent vector, and $Q \in \mathbb{R}^3$. Since the tangent is constant along a line, we can take a constant frame $\mathbf{U} = [\mathbf{u}\ \mathbf{v}\ \mathbf{w}]$. For the level value $c \in (0,1)$, let ω_c be the only solution of $\omega_c - \omega_c^3 + \frac{3}{5}\omega_c^5 - \frac{1}{7}\omega_c^7 = \frac{16}{35}(1-c)$ in $(0,1)$, and let $\eta_c = \sqrt{1-(\frac{c}{2})^{\frac{2}{7}}}$. The next proposition gives the values for the eigenvalues that achieve precise radii on each direction around Γ. (See supplementary material for proof and derivations of the formulas for ω_c and η_c).

Proposition 5.3.2 (Radii control)**.** *Given the radii r_u, r_v, r_w such that $\frac{r_u}{\omega_c}(\frac{c}{2})^{\frac{1}{7}} \leq \frac{l}{2}$, let $\alpha = \frac{\omega_c^2}{r_u^2}$, $\beta = \frac{\eta_c^2}{r_v^2}$ and $\gamma = \frac{\eta_c^2}{r_w^2}$. Then $\{Q - r_u\mathbf{u}, Q + \hat{r}_u\mathbf{u} + r_v\mathbf{v}, Q + \hat{r}_u\mathbf{u} + r_w\mathbf{w}\} \subset \mathscr{S}_\Sigma^c$ for $\hat{r}_u \in [\frac{r_u}{\omega_c}(\frac{c}{2})^{\frac{1}{7}}, l - \frac{r_u}{\omega_c}(\frac{c}{2})^{\frac{1}{7}}]$.*

Proof. Notice that $p'(x) = (1-x^2)^3$, hence $\int_0^x K(t)\mathrm{d}t = \frac{35}{16}p(x)$ for $0 \leq x \leq 1$. Also, $p(\omega_c) = \frac{16}{35}(1-c)$ implies that $\frac{d\omega_c}{dc} = -\frac{16}{35}\frac{1}{(1-\omega_c^2)^3} < 0$. Now

$$C^K_{\Gamma,\mathbf{G}}(Q - r_u\mathbf{u}) = \int_0^l K\left(\sqrt{(s+r_u)^2\alpha}\right)\sqrt{\alpha}\,\mathrm{d}s$$

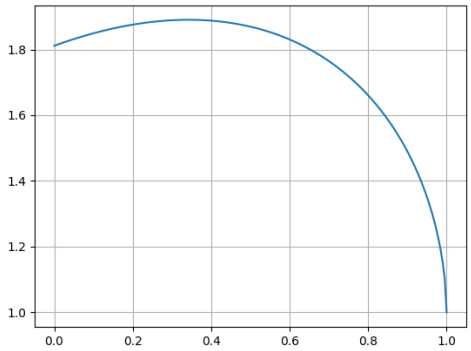


Figure 5.11: Plot of $2\left(\frac{1-\frac{35}{16}p(\omega_c)}{2}\right)^{\frac{1}{7}}+\omega_c$, for $\omega_c \in (0,1)$.

$$
\begin{aligned}
&= \int_{r_u\sqrt{\alpha}}^{(l+r_u)\sqrt{\alpha}} K(t)\,\mathrm{d}t \quad [\text{taking } t=(s+r_u)\sqrt{\alpha}] \\
&= \int_{0}^{(l+r_u)\sqrt{\alpha}} K(t)\,\mathrm{d}t - \int_{0}^{r_u\sqrt{\alpha}} K(t)\,\mathrm{d}t \\
&= 1-\frac{35}{16}p(r_u\sqrt{\alpha}) \quad [\text{using } (l+r_u)\sqrt{\alpha}\geq 1] \\
&= 1-\frac{35}{16}p(\omega_c) = c
\end{aligned}
$$

To see that $(l+r_u)\sqrt{\alpha}\geq 1$, notice that $(l+r_u)\sqrt{\alpha}\geq 2\left(\frac{c}{2}\right)^{\frac{1}{7}}+\omega_c \geq 1$. The last inequality can be proved formally, with a long technical development. It is easier to see the numerical plot of $2\left(\frac{c}{2}\right)^{\frac{1}{7}}+\omega_c$ for $\omega_c \in (0,1)$ (See Figure 5.11).

$$
\begin{aligned}
C^K_{\Gamma,\mathbf{G}}(Q+\hat{r}_u\mathbf{u}+r_v\mathbf{v}) &= \int_0^l K\left(\sqrt{(s-\hat{r}_u)^2\alpha+r_v^2\beta}\right)\sqrt{\alpha}\,\mathrm{d}s \\
&= \int_{-\hat{r}_u\sqrt{\alpha}}^{(l-\hat{r}_u)\sqrt{\alpha}} K\left(\sqrt{t^2+\eta_c^2}\right)\mathrm{d}t \quad [\text{taking } t=(s-\hat{r}_u)\sqrt{\alpha}] \\
&= \int_{-\hat{r}_u\sqrt{\alpha}}^{(l-\hat{r}_u)\sqrt{\alpha}} K\left(\sqrt{1+t^2-\left(\frac{c}{2}\right)^{\frac{2}{7}}}\right)\mathrm{d}t \\
&= \int_{-\left(\frac{c}{2}\right)^{\frac{1}{7}}}^{\left(\frac{c}{2}\right)^{\frac{1}{7}}} K\left(\sqrt{1+t^2-\left(\frac{c}{2}\right)^{\frac{2}{7}}}\right)\mathrm{d}t \\
&\left[\text{since } l-\hat{r}_u\sqrt{\alpha}\geq \hat{r}_u\sqrt{\alpha}\geq\left(\frac{c}{2}\right)^{\frac{1}{7}} \text{ and } K(\cdot) \text{ is zero for } |t|>\left(\frac{c}{2}\right)^{\frac{1}{7}}\right]
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{35}{8}\int_0^{\left(\frac{c}{2}\right)^{\frac{1}{7}}} \left(\left(\frac{c}{2}\right)^{\frac{2}{7}} - t^2\right)^3 \mathrm{d}t \\
&= \frac{35}{8}\int_0^{y} (y^3 - 3y^2t^2 + 3yt^4 - t^6)\,\mathrm{d}t \quad \left[\text{for } y = \left(\frac{c}{2}\right)^{\frac{1}{7}}\right] \\
&= \frac{35}{8}\left[y^6t - y^4t^3 + \frac{3}{5}y^2t^5 - \frac{1}{7}t^7\right]_0^y = \frac{35}{8}\left(y^7 - y^7 + \frac{3}{5}y^7 - \frac{1}{7}y^7\right) \\
&= \frac{35}{8}\frac{16}{35}y^7 = c
\end{aligned}
$$

Analogously

$$C^K_{\Gamma,\mathbf{G}}(Q + \hat{r}_u\mathbf{u} + r_w\mathbf{w}) = c \qquad \square$$

From Proposition 5.3.2 we get several important conclusions. First, we can achieve precise control over the radius in the tangent direction (tip distances). Second, each eigenvalue is defined by c (the level value) and the desired radius (r_u, r_v, r_w), independently of the other two eigenvalues, i.e. changing one radius does not impact the others. Third, the radii in the normal directions are guaranteed in the interval $[\frac{r_u}{\omega_c}(\frac{c}{2})^{\frac{1}{7}}, l - \frac{r_u}{\omega_c}(\frac{c}{2})^{\frac{1}{7}}]$, depending only on the radius in the tangent direction and c. And last: by choosing a level set close to zero we can achieve the radii in the normal directions sufficiently close to the tips, within a valid interval along the curve, i.e. such that $\frac{r_u}{\omega_c}(\frac{c}{2})^{\frac{1}{7}} \leq \frac{l}{2}$. This follows from $\frac{1}{\omega_c}(\frac{c}{2})^{\frac{1}{7}}$ being an increasing function of c with range $[0, +\infty]$ for $c \in [0,1]$. For $c = 0.1$, $\frac{1}{\omega_c}(\frac{c}{2})^{\frac{1}{7}} \approx 1.18654$. Note though, that choosing a particular level set $\tilde{c}$ for one skeletal curve $\Gamma \in \Sigma$ is equivalent to change the constant δ_Γ to $\frac{c}{\tilde{c}}\delta_\Gamma$ in Equation (5.3.2). This way the overall convolution surface is still defined by the original level value c.

Thanks to Proposition 5.3.2, the user only has to define radii at the end-points of the skeletal curves, the eigen-values $\alpha_i, \beta_i, \gamma_i$ are then computed automatically. We perform eigen-value interpolation along the skeletal curves such that the radii of the metric matrix ellipsoid vary linearly. This is done using the following formula:

$$\chi(s) = \left(\frac{l-s}{l}\chi_0^{-\frac{1}{2}} + \frac{s}{l}\chi_1^{-\frac{1}{2}}\right)^{-2} \quad \text{for } \chi = \alpha, \beta, \gamma. \tag{5.3.10}$$

Figure 5.12 shows examples of radii control along the shape. We used Proposition 5.3.2 to compute the associated eigen-values given some desired radii at each extremity of Γ. Notice that, although Proposition 5.3.2 assumes Γ to be a line segment and the radii to be constant, the eigenvalues computed with the same method works well for varying radii around both line segments and arcs of circle.

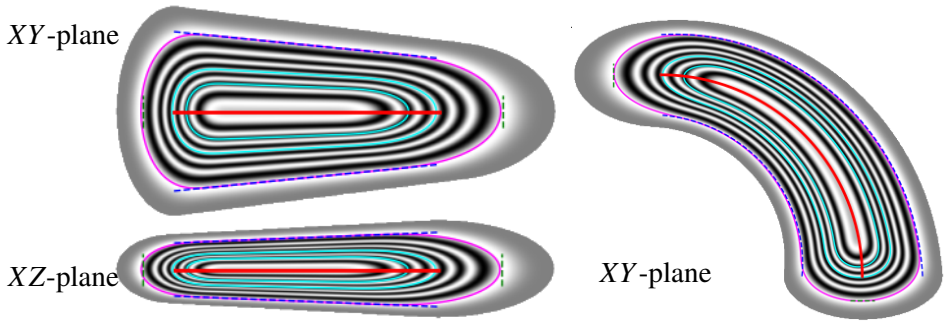
Proposition 5.3.3 (Ellipsoidal cross section)**.** *Let $r_u, r_v, r_w, \hat{r}_u, \alpha, \beta, \gamma$ be as in Proposition 5.3.2. Then for every $\mu, \xi \in \mathbb{R}$ such that $\frac{\mu^2}{r_v^2} + \frac{\xi^2}{r_w^2} = 1$ we have that $(Q + \hat{r}_u\mathbf{u} + \mu\mathbf{v} + \xi\mathbf{w}) \in \mathscr{S}^c_\Sigma$.*

Proof.

$$C^K_{\Gamma,\mathbf{G}}(Q+\hat{r}_u\mathbf{u}+\mu\mathbf{v}+\xi\mathbf{w}) = \int_0^l K\left(\sqrt{(s-\hat{r}_u)^2\alpha+\mu^2\beta+\xi^2\gamma}\right)\sqrt{\alpha}\,\mathrm{d}s$$
$$= \int_{-\hat{r}_u\sqrt{\alpha}}^{(l-\hat{r}_u)\sqrt{\alpha}} K\left(\sqrt{t^2+\eta_c^2}\right)\mathrm{d}t.$$

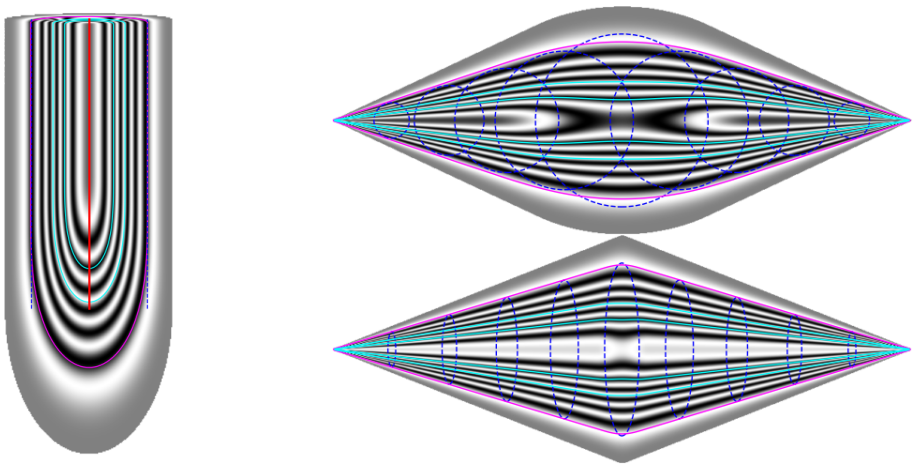
From here we continue as in the proof of Proposition 5.3.2 □

As already illustrated in Figure 5.12, anisotropic convolution provides not only an ellipse-like normal section but also caps of independent depth at the extremities. Such a desirable feature would require tweaking the skeleton tips in [Zanni et al., 2013]. If as in previous convolution formalism all radii are equal ($\alpha = \beta = \gamma$), the convolution can be seen as smoothing a union of sphere centered on the skeleton. On the other end, if $\alpha \ll \beta, \gamma$, anisotropic convolution can be seen as smoothing of a union of discs normal to the skeleton (see Figure 5.13b). Even for circular normal sections ($\beta = \gamma$), this property allows to model shape features not amenable to previous convolution formalisms. The shape can easily be tapered at the extremities (Figure 5.13a) and its cross-section can vary more rapidly along the skeleton.



(a) Line segment with varying radii. (b) Arc of circle with varying radii.

Figure 5.12: Radii control in anisotropic convolution. We show the level sets for $c = 0.1$ (magenta), $c = 1.0$ and $c = 1.5$ (cyan). The blue dashed line shows the linear interpolation of the user-defined radii: from 1.5 to 1.0 in the XY-plane, from 0.7 to 0.5 in the XZ-plane. The green dashed line shows the expected radii in the tangent direction at extremities: from 0.6 to 1.2. Notice how the surface level set ($c = 0.1$) is very close to the expected radii interpolation even in the case of arcs of circle. In gray-scale we show the values of the convolution field composed with $x \to \sin(20x)$. The picture illustrates the boundary outside which the field value is zero. In red we show the skeletal curve.



(a) Varying α with β, γ constant.

(b) Varying α, β, γ.

Figure 5.13: Impact of tangential radii on the surface. Variation of α is only visible at endpoints (left). If the tangential radii is equal to orthogonal radii then the shape is similar to a union of sphere (right top) else if α tend toward zero, the shape is similar to a union of discs (right bottom).

5.4 Skeleton-driven meshing

In this section we introduce a meshing technique for convolution surfaces that exploits the skeleton structure. Our approach produces quad-dominant meshes that follow the skeleton, providing a good edge-flow, whether coarse or refined. We first compute a *scaffold* [Panotopoulou et al., 2018, Fuentes Suárez and Hubert, 2018b] – a coarse quad mesh around the skeleton – adapted to $\mathscr{G}^1$ curves, that is then projected onto the surface. The mesh so obtained reflects the smoothness of the surface without post-processing. With this approach the evaluation of the convolution field is delayed until the projection step, which is done only once for each point in the scaffold. Our meshing technique is intended for surfaces that reflect the topology, and to some extent the geometry, of the skeleton.

Common choices for polygonization of implicit surfaces, Marching Cubes [Lorensen and Cline, 1987] and variants [Gomes et al., 2009, Chapter 7], produce triangle meshes that may misrepresent the topology of the surface for coarse grids. We produce quad meshes with accurate topology and good edge-flow, independently of the coarseness of the mesh. With quads we can exploit the curvature behavior along the skeleton, having longer side-length along the skeleton (where the curvature is lower) than transversally (higher curvature). Our projection step also guarantees that the mesh has all its vertices *on* the surface.

In [Fuentes Suárez and Hubert, 2018b] a *scaffold* is a quad mesh built around an articulated skeleton made of line segments. Among other scaffolds-like constructions [Panotopoulou et al., 2018, Ji et al., 2010, Bærentzen et al., 2012,

Usai et al., 2015, Yao et al., 2009], [Fuentes Suárez and Hubert, 2018b] works for skeletons of any topology, including skeletons with cycles like in Figure 5.17, without introducing extra quads at the joints. It also provides scaffolds with the same number of quads around each line segment, and scaffolds respecting the symmetries of the skeleton. [Fuentes Suárez and Hubert, 2018b] constructs polygonal "tubes" around each line segment that meet at the joints. The "tubes" intersect the unit sphere centered at the joint creating a partition into convex spherical polygons. The number of vertices and edges on the spherical polygons depend on the topology and the geometry of the whole skeleton. The spherical polygons define a set of unit vectors at each extremity of the line segments that are *connected* by the quads (Figure 5.14).

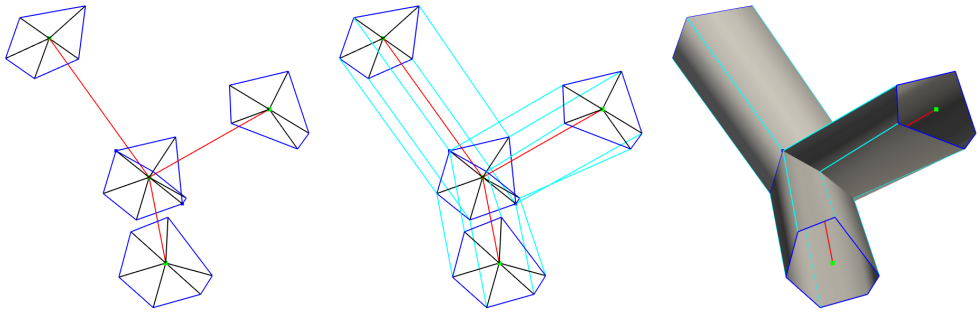


Figure 5.14: The scaffolding method constructs a quad mesh (right) from a line segment skeleton (red lines). This is done by constructing unit vectors (black lines) around each line segment extremities (green) that are then connected (cyan lines) in a bijective way. The vectors connected along each line segment define a polygon (blue) that encloses the corresponding segment at each extremity. Notice that some vectors (like in the middle joint) are connected along two or more line segments.

5.4.1 Scaffolds for circular splines

The scaffold in [Fuentes Suárez and Hubert, 2018b] is limited to line segment skeletons. To use it on circular spline skeletons we first take, for each skeletal curve, the polyline representation given by the collection of biarc tangents (Figure 5.2). We then modify the *connections* in the output of [Fuentes Suárez and Hubert, 2018b]: we drop the intermediate vectors of the polyline and connect directly the vectors at the endpoints of the skeletal curves (Figure 5.15). The scaffold is constructed around the topologically relevant skeletal curves. For example, in *soft carving* the pieces of the skeleton with $\delta_\Gamma < 0$ only provide details on surface and are not to support any pieces of the scaffold.

For a skeletal curve $\Gamma([0,l]) \in \Sigma$, with frame $\mathbf{F}([0,l])$: let m be the number of quads we want along Γ; let $\{\mathbf{f}_0^j\}_{j=0}^k$ and $\{\mathbf{f}_m^j\}_{j=0}^k$, be the sets of $k+1$ vectors computed by the scaffolding algorithm at $\Gamma(0)$ and $\Gamma(l)$ respectively. The indices j represent the

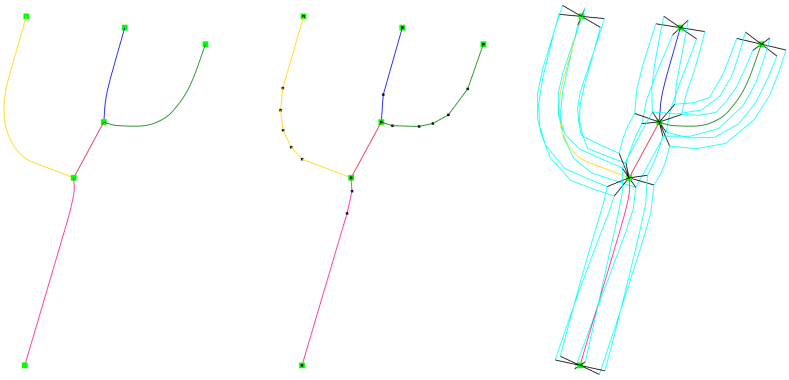


Figure 5.15: Meshing with a scaffold. From left to right: circular spline skeleton, tangential polyline discretization of the skeletal curves, and connection (cyan) of the vectors (black) along the curves. Notice that we *connect* directly the vectors at the endpoints of each curve

counterclockwise order, according to the frames $\mathbf{F}(0)$ and $\mathbf{F}(l)$ respectively, on the polygon generated around the extremities of Γ.

We connect the vectors by minimizing the rotation within the frame $\mathbf{F}$ along Γ. That is, we connect $\mathbf{f}_0^j$ with $\mathbf{f}_m^{j+q}$ where

$$q = \underset{w=0\ldots k-1}{\operatorname{argmin}} \sum_{j=0}^{k} \|(\mathbf{F}(0))^{\mathsf{T}}\mathbf{f}_0^j - (\mathbf{F}(l))^{\mathsf{T}}\mathbf{f}_m^{k-j+w}\| \tag{5.4.1}$$

and $k-j+w$ is taken modulo k. For simplicity we relabel the vectors such that $\mathbf{f}_0^j$ is connected with $\mathbf{f}_m^j$.

To generate the quads along the skeletal curve we *transport* the vectors with varying local coordinates. Formally, given two unit vectors $\mathbf{f}_0, \mathbf{f}_m$ positioned at $\Gamma(0)$ and $\Gamma(l)$ respectively, we define the *transport* of $\mathbf{f}_0$ to $\mathbf{f}_m$ as

$$\mathbf{f}(t) = \mathbf{F}(t)\frac{\bar{\mathbf{f}}(t)}{\|\bar{\mathbf{f}}(t)\|}, \tag{5.4.2}$$

where $\bar{\mathbf{f}}$ is an interpolation from $\mathbf{F}(0)^{\mathsf{T}}\mathbf{f}_0$ to $\mathbf{F}(l)^{\mathsf{T}}\mathbf{f}_m$ that avoids the origin. In our setting we use the linear interpolation $\bar{\mathbf{f}}(t) = \frac{l-t}{l}\mathbf{F}(0)^{\mathsf{T}}\mathbf{f}_0 + \frac{t}{l}\mathbf{F}(l)^{\mathsf{T}}\mathbf{f}_m$ (Figure 5.16), valid when $\mathbf{F}(0)^{\mathsf{T}}\mathbf{f}_0 + \mathbf{F}(l)^{\mathsf{T}}\mathbf{f}_m \neq 0$.

We then compute the *mesh lines* $\{\mathbf{f}_i^j\}_{i=0}^m$ $(j = 0,\ldots,k)$ where $\mathbf{f}_i^j = \mathbf{f}^j\left(\frac{i}{m}l\right)$ with $\mathbf{f}^j$ defined by the *transport* of $\mathbf{f}_0^j$ to $\mathbf{f}_m^j$ (5.4.2). The vertices $P_i^j = \Gamma\left(\frac{i}{m}l\right) + \mathbf{f}_i^j$ then define the quads

$$\mathcal{Q}_i^j = (P_i^j, P_{i+1}^j, P_{i+1}^{j+1}, P_i^{j+1}) \quad i = 0,1,\ldots,m-1 \quad j = 0,1,\ldots,k,$$

where the indices are considered modulo k.

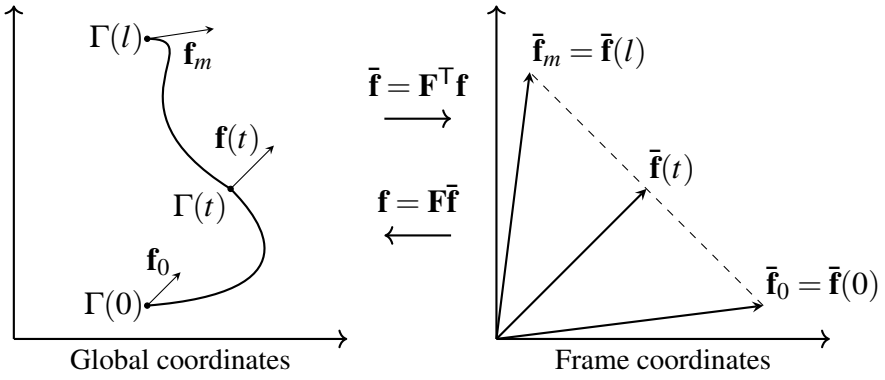


Figure 5.16: Transporting $\mathbf{f}_0$ to $\mathbf{f}_m$ along the curve Γ with frame $\mathbf{F}$.

To close the mesh at the tips one can choose to use polar-annular caps [Bærentzen et al., 2014], which includes degenerate quads around a high valency irregular vertex. Or quad layout that includes several irregular vertices but with lower valency [Wu and Liu, 2012, Section 5.3]. In practice we use polar-annular caps.

5.4.2 Mesh projection onto the convolution surface

To obtain the final polygonization we project the mesh described in the previous section. The projection is done by ray-shooting from the skeleton onto the surface.

Given a point $T \in \Gamma([0,l])$, and a direction $\mathbf{u}$ with $\|\mathbf{u}\| = 1$, we find the first intersection of the ray $r(t) = T + t\mathbf{u}$ with the convolution surface around Γ. This means finding the minimal solution $t > 0$ of

$$C^K_{\Gamma,\mathbf{G}}(T + t\mathbf{u}) = c. \tag{5.4.3}$$

By definition, the point $P = T + t\mathbf{u}$ is on the convolution surface of the level set $c > 0$, for any $t > 0$ satisfying (5.4.3). We can exploit the decreasing nature of the field value when moving away from the skeleton to efficiently compute a solution of (5.4.3). Moreover, for compact support kernels the solutions can be bounded in an interval computed from the radius of the kernel. In our implementation we solve (5.4.3) with Brent's method [Brent, 2002] (available in GSL).

The quality of the obtained quad mesh can be appreciated in the wire-frames in Figures 5.10a, 5.10c, 5.17 and 5.20. In Figure 5.17 we show high-genus surface meshes.

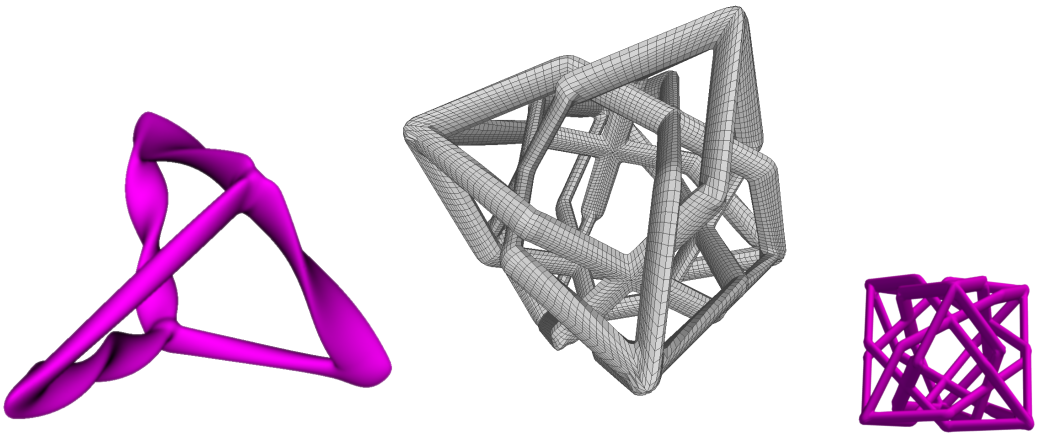


Figure 5.17: Abstract cage-like surfaces of positive genus (skeletons with cycles).

5.5 Implementation and Numerical integration

The meshing algorithm was prototyped in Python with call to a C library for field evaluation. Since the anisotropic convolution field requires a frame at each point of the skeletal curve, obtaining closed form formulas for anisotropic convolution (Equation (5.3.5)) is unrealistic. On the other hand quadrature methods for numerical integration provide accurate results up to a prescribed tolerance [Piessens et al., 1983]. In our implementation we use the QUADPACK family of quadrature methods of the GNU Scientific Library (GSL) [Galassi et al., 2017]. Each integral is evaluated using the *QAG adaptive integration method* [Galassi et al., 2017] with 61 Gauss-Kronrod quadrature points, limited to a maximum of 100 subintervals, and absolute error of 10^{-8}. When the radii are close to zero at one extremity, like in Figure 5.13b, the numerical accuracy of the field evaluation at a point $X \in \mathbb{R}^3$ is improved by placing more quadrature points in the vicinity of the closest point to X on the skeleton. This can be effectively achieved by splitting the integral evaluation at the closest point of the skeleton into two integrals.

On all examples, timings are driven by the ray-shooting phase of the algorithm which is proportional to the number of vertices in the resulting mesh. For instance, in Figure 5.1a, the meshing time is 28.3s for the surface around the arcs of circle approximation, and 30.2s around line segments (using 4 core of an Intel Core i7-6600U CPU @ 2.60GHz). The total number of vertices in the mesh is 1160. The number of integral evaluations is more than 250k in both cases, with arcs of circle having in average less field evaluations per ray-shooting (22.75) than line segments (23.7). The average integral evaluation time is 0.34ms for an arc of circle, and 0.33ms for a line segment. Those numbers highlight two facts. First, it confirms that it is computationally more interesting to use circles for curve approximation. Secondly, an optimized vertex projection implemented on the GPU [Zhu et al., 2011, Zhu et al., 2015a] would improve

timings.

5.6 Examples & Applications

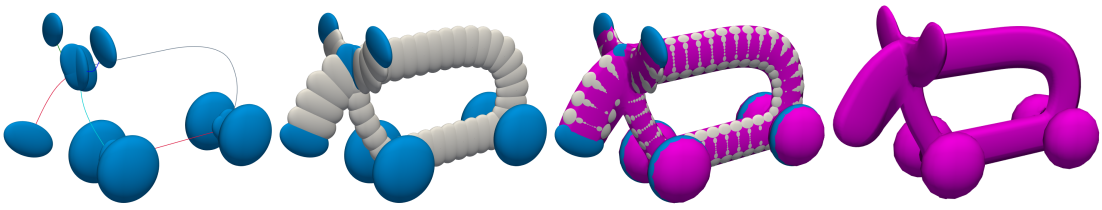
We illustrate the new capabilities of anisotropic convolution through three example applications: skeleton-based modeling, more general implicit modeling, and surface approximation.

5.6.1 Skeleton-based modeling

As with previous convolution surfaces, skeleton-based modeling is a direct application of our new formulation. The user only needs to define the circular splines with ellipsoids at the end-points. The current implementation does not provide real-time generation of the surface. However, it should be noted that due to the properties of anisotropic convolution surfaces, the implicit surface is close to the infinite union of interpolated ellipsoids along the splines. A dense enough subset of those ellipsoids can therefore provide a real-time preview to guide the user during modeling (see Figure 5.18).

In Figure 5.19 we show an example of surface meshes around a closed curve. The anisotropic surfaces are defined on a circular spline approximation of the smooth curve. We showcase the correction angle needed for RMFs around closed curves (sections 5.2.2 and 5.3.2).

In Figure 5.20 we show a salamander model. Anisotropy, tangential radii variation and RMF were instrumental for the tail and head of the salamander. We also show a wire-frame representation of the quad mesh.



(a) User inputs. (b) Interpolation. (c) Convolution. (d) Surface.

Figure 5.18: Elk model (10 biarcs, 15 line segments). Note how the interpolating ellipsoids provide an accurate preliminary visualization for the final convolution surface.

5.6.2 Blobtree modeling

We also used the new formulation in a more general context, namely *Blobtree modeling* [Wyvill et al., 1999]. We follow the adequate inner bound methodol-

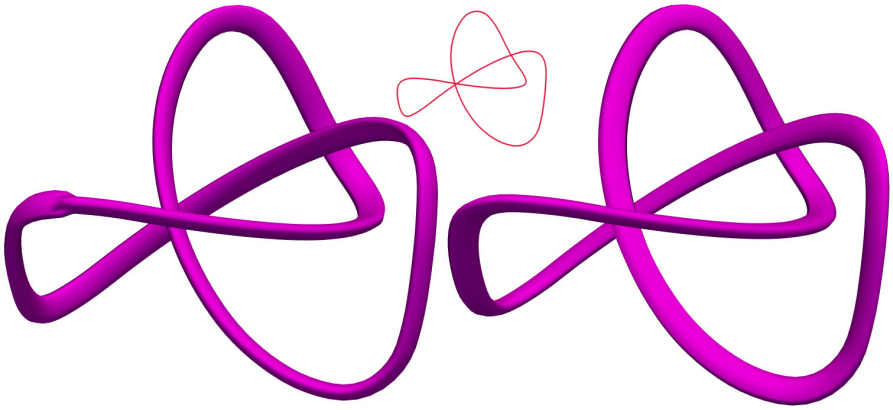


Figure 5.19: Convolution surface around a circular spline approximation of the knot $\Gamma(t) = (-10\cos t - 2\cos 5t + 15\sin 2t, -15\cos 2t + 10\sin t - 2\sin 5t, 10\cos 3t)$, $t \in [0, 2\pi]$. Anisotropic convolution parameters: $\alpha_0 = \alpha_1 = \gamma_0 = \gamma_1 = 1$, $\beta_0 = \beta_1 = 1/4$. On the left the surface without rotation: $\theta_0 = \theta_1 = 0$, on the right the surface with a corrective rotation: $\theta_0 = 0$ and $\theta_1 = -1.89$. In the middle we show the skeleton. We used 19 biarcs to approximate Γ.

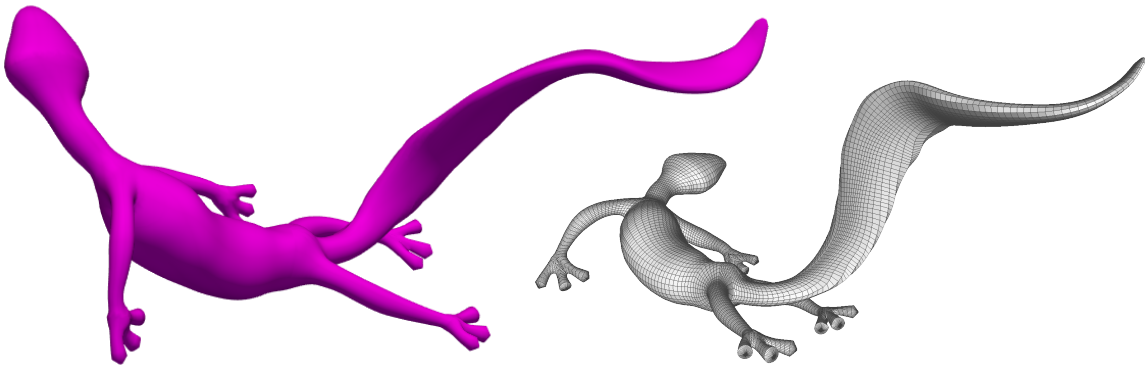


Figure 5.20: Salamander model (45 biarcs, 48 line segments). Left: surface computed with our meshing technique. Right: wire-frame showing the quads of the mesh.

ogy [Canezin et al., 2013] in order to define a smooth carving operator from Ricci's blending operator [Ricci, 1973]. For this purpose, we used the transfer function proposed in [Ricci, 1973] to re-map field value to the expected range ($[0, 1]$ with 0.5 as level value). Figure 5.21 depicts a cup modeled with this approach. All primitives used in this object rely on anisotropy, and noticeably on the anisotropy in the tangential direction. Note that when a large anisotropy in the tangent direction is used (as done on the foot of the cup) the maximal amount of blending that could be achieved by subsequent operations on the scalar field is reduced. We did not rely on our meshing approach for this object due to the large difference between the skeleton and the medial axis of the final surface (mainly due to the carving used to hollow the cup).

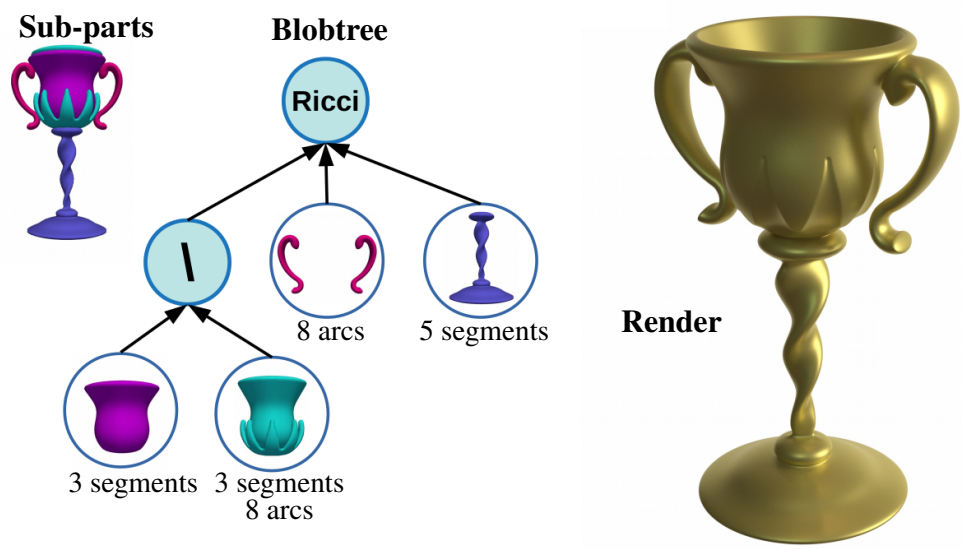


Figure 5.21: A cup model. Left: the *blobtree* used to design the cup, 27 primitives where used in total, including 11 for the carving. Right: rendering of a mesh extracted with Marching Cubes. Note the relevance of twisting and anisotropy, both in the normal and tangent directions, to model the different smooth parts by our new convolution with few parameters.

5.6.3 Shape approximation

Finally, we use anisotropic convolution in order to compute semi-automatically smooth and compact representation of surfaces representing volumes.

The steps of our approximation method are:

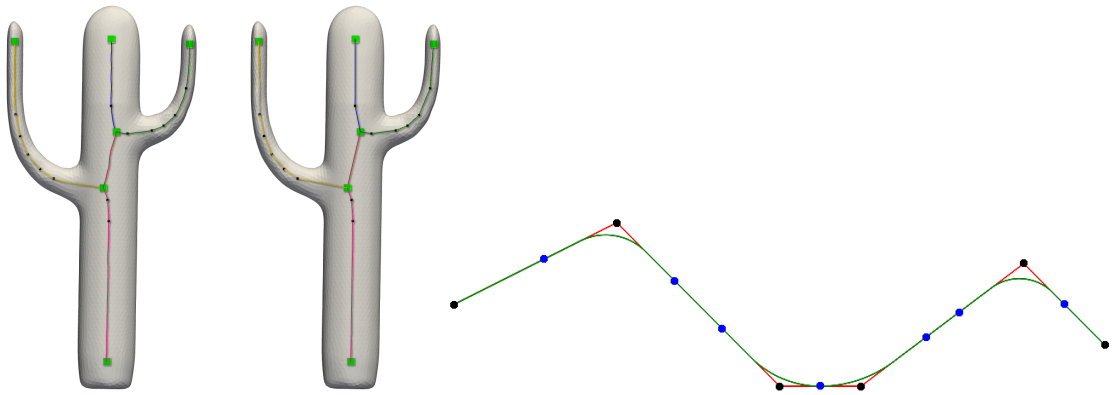
1. Compute skeleton consisting of short line segments.

2. Identify *branches* on the skeleton.

3. Simplify and approximate each branch.

4. Optimize the parameters of anisotropic convolution on each branch.

The first three steps give the set of curves defining the skeleton and are described in Section 5.6.3. The last step, computing the parameters along the skeletons, is described in Section 5.6.3.



(a) Branch simplification. (b) Circular spline from polyline.

Figure 5.22: Post-processing of skeletonization algorithms output. In picture (a), on the left, the output of a skeletonization algorithm; on the right, the simplified branch polylines. We highlight the branches of the skeleton in different colors. In black, we show the points selected as vertices of the simplified polyline. The *branching points* appear in bright green. In picture (b), we show the tangential circular spline, in green, obtained from the red polyline. The original points of the polyline are shown in black; in blue, the tangential points.

Skeleton approximation

Given the output $\mathscr{O}$ of a skeletonization algorithm, we define a *branch* as the portion of $\mathscr{O}$ connecting two *branching points*. In general a point of $\mathscr{O}$ is considered a *branching point* if it is connected with either only one or more than two other points in the skeleton. A point in $\mathscr{O}$ is identified as additional branching point if: a sudden change of distance from the model surface to the skeleton is detected, or the incident line segments have a sharp angle ($< \pi/2$). Some extra branching points may be identified by the user to better reflect features of the input model. To reduce the number of line segments, we simplify each branch with a polyline simplification algorithm [Douglas and Peucker, 1973] (Figure 5.22a). When the tips of the branches are not sufficiently close to the input model (along tangent direction), we extend the skeletons. Additional manual editing of the skeleton can be performed by the user (as illustrated in Figure 5.23). The collection

of simplified polyline branches is denoted by $\mathscr{B}$. For each polyline $B \in \mathscr{B}$ we compute a tangential circular spline approximation Γ_B that interpolates the endpoints of B. Figure 5.22b illustrates this process.

Parameters optimization along circular spline

To fit an anisotropic convolution on each spline (Step 4) we first associate each vertex in the input mesh with its closest branch. For each branch $B \in \mathscr{B}$, $\mathscr{P}_B$ denotes the set of the associated vertices.

Then we perform a non-linear least square optimization [Gomes et al., 2009, Chapter 8] of the anisotropic convolution field (5.3.5). For each circular spline Γ_B ($B \in \mathscr{B}$) we look for the parameters $\alpha_i, \beta_i, \gamma_i, \theta_i$ ($i = 0, 1$) of its corresponding anisotropic convolution field $C^K_{\Gamma_B}$ that minimize

$$\sum_{P \in \mathscr{P}_B} \left(w(C^K_{\Gamma_B, \mathbf{G}}(P)) - c \right)^2. \tag{5.6.1}$$

With $w(\cdot)$ a weighting function that rescales values outside the convolution surface. This is needed because field values inside the convolution function are in the range $(c, 2]$, while outside the range is $[0, c)$. For $c = 0.1$ (chosen level set) this would give an imbalance on the least squares optimization. The general expression for w is

$$w(x) = \begin{cases} W(x - c) + c & 0 \leq x \leq c \\ x & \text{otherwise} \end{cases} \tag{5.6.2}$$

where $W \in \mathbb{R}$ is a constant that controls the additional weight for points outside the convolution surface. Experimental results show that a value of W between 3 and 5 gives better fits for our chosen kernel (Section 5.3) and level set ($c = 0.1$).

In order to attain a valid solution the optimization parameters in (5.6.1) must be bounded. In practice we take $\alpha_i = 1$, $-\frac{\pi}{2} \leq \theta_i \leq \frac{\pi}{2}$, $0 < \beta_i, \gamma_i < e_B$, with e_B taken such that all the points $P \in \mathscr{P}_B$ are inside the convolution surface defined by $\alpha_i = 1, \beta_i = \gamma_i = e_B, \theta_i = 0$ on the level set c. These values are also taken as the starting point in the optimization. They guarantee that the starting field is nonzero on every point $P \in \mathscr{P}_B$, i.e. all the points are inside the surface. For compact support kernels this is required to secure a non-zero gradient: outside the support the field is constantly zero.

Approximation experiments and discussion

In Figure 5.23 we show a semi-automatic anisotropic fitting resembling the fertility model. In this experiment the user modified the circular spline skeleton obtained from the output of a skeletonization algorithm (Section 5.6.3). Then the parameters defining the convolution surfaces were computed with our optimization step (Section 5.6.3).

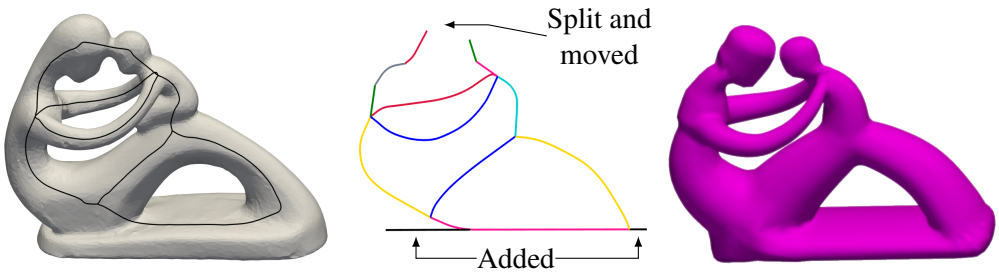


Figure 5.23: Semi-automatic approximation of the fertility model by anisotropic convolution surface. Left: Fertility model and output (black polyline) of the skeletonization algorithm in [Yan et al., 2016]. Middle: Circular splines after manual editing - adding the base (black lines) and splitting the top (red and green splines). Right: Result of the anisotropic convolution fitting process.

In Figure 5.24 we compare the results of automatic fittings to a cactus model with three skeletonization methods. The fitting is done on the original model, a randomly decimated model, and a model with holes. In the decimated models we reduced the number or points on the mesh to one fifth, reducing the computational time of the optimization accordingly. The original model has around 5200 vertices. In terms of Hausdorff distance, it is noticeable how the decimated model results in as good approximation as the original fitting. Similarly, for incomplete models (with holes) the fitting performs well.

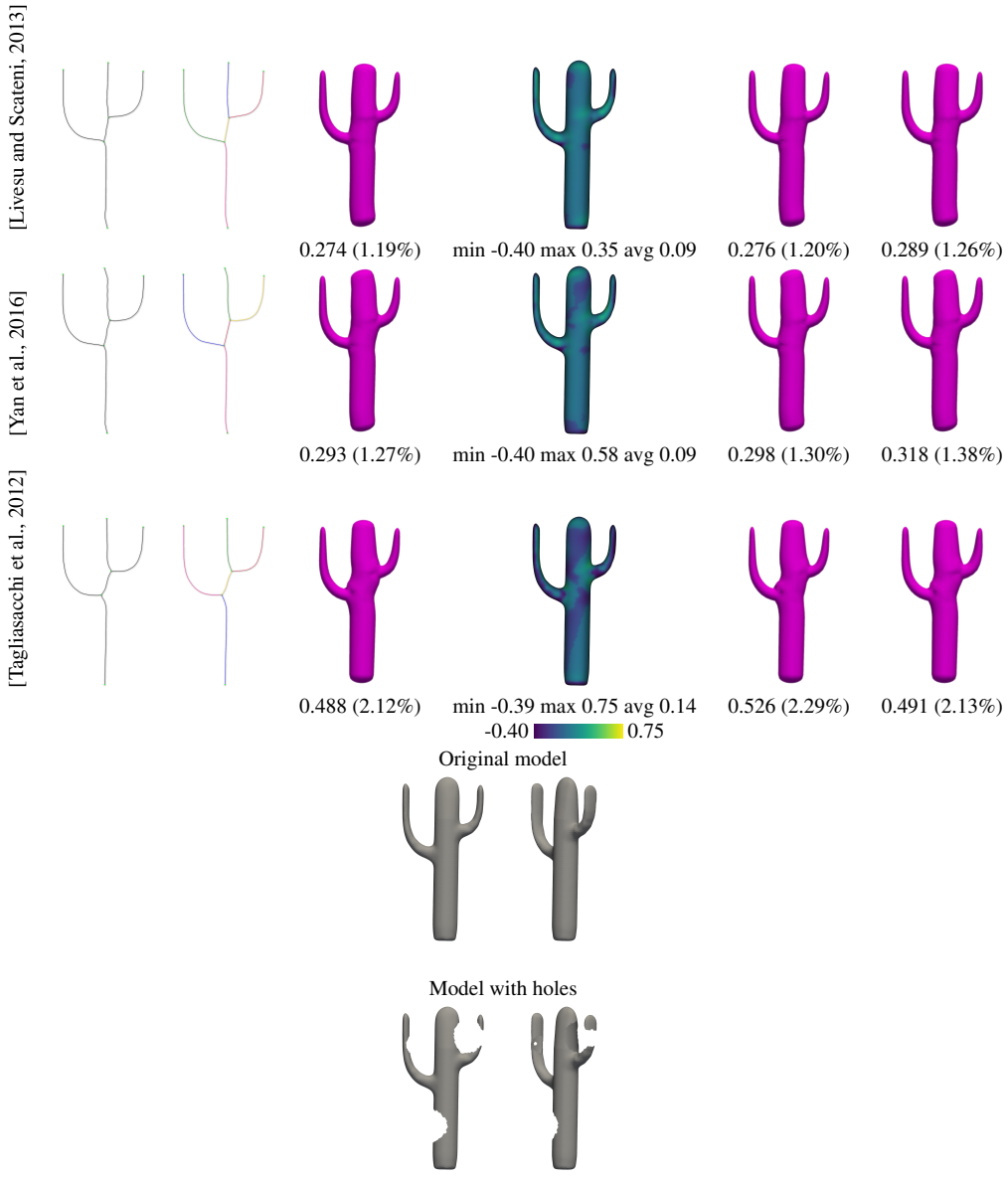


Figure 5.24: Surface approximation via skeletonization. On the left we show the output of the fitting process for the models on the right. Columns, from left to right: output of skeletonization algorithms, circular spline skeletons, fitted surfaces, weighted field value deviations from the level value ($c = 0.1$), fitted surface on a decimated model (one-fifth of points), fitted surface on a decimated model (one-fifth) with holes. Below the fitted surfaces we show the Hausdorff distances to the original model, in magnitude and as a percent of the bounding box diagonal (23.0 units).

5.7 Conclusions

We have extended the modeling capabilities of convolution surfaces with a method that controls the anisotropy of the surface around the skeleton. For $\mathscr{G}^1$ skeletal curves we introduced the use of biarcs approximation in order to use less pieces, while still retaining $\mathscr{G}^1$ continuity for the skeleton and smoothness in the final surface. The $\mathscr{G}^1$ skeleton allows to define a shape along the curve that is intuitive and easy to model, with few parameters that guarantee control over the radii around the skeleton. We developed a meshing technique for convolutions surfaces based on a scaffolding technique that generates a quad-dominant mesh that follows the structure of the skeleton. The variety of shapes and new modeling freedom is showcased in several examples. Mimicking shapes generated with 2D skeletons is another advantage of the new formulation. We also presented applications of anisotropic convolution to skeleton-based modeling, general implicit modeling, and surface approximation.

Part III

Implementation

Chapter 6

PySkelton: scaffolding and anisotropic convolution in Python

The methods and algorithms we discuss in this work have been implemented in a Python package: **PySkelton**[1]. This library can be used to construct (symmetric regular) scaffolds for skeletons of any topology (Chapter 1). It can also be used to model anisotropic convolution surfaces and output quad-dominant meshes of the surfaces (Chapter 5). Additionally the library is capable of using spherical Laguerre diagrams for the scaffolding, and of generating hexahedral meshes (Chapter 2). We designed **PySkelton** to be academic-friendly and extensible. In this chapter we present the general design of the library and some of the implementation challenges. Our purpose is twofold: this chapter will serve as documentation of the library, and as illustration of issues related to the implementation of scaffolding and anisotropic convolution surfaces.

In this chapter we describe the classes of **PySkelton**, provide examples of use, and discuss some implementation details.

6.1 General design

When using PySkelton the user needs to define first a skeleton, then a surface, and finally generate a quad-dominant mesh. The typical use follows three steps:

1. Define a skeleton and associate some connectivity information in form of a graph.

   In this step the user creates one or more instances of the abstract class `Skeleton` (through derived classes). The connectivity information is defined as a graph given by an instance of the `Graph` class.

2. Define an anisotropic convolution surface by setting up the convolution field parameters for each skeleton piece.

   At this point the user creates an anisotropic convolution field for each skeleton piece and provides the required parameters (see Section 5.3.2) in the constructor of classes derived from `Field`.

3. Generate a quad-dominant mesh for the surface or scaffold and visualize it.

   The mesh generation is done with the classes `Scaffolder` and `Mesher`, while the visualization is handled by classes derived from `Visualization`.

Step 2 is optional for users interested only in a scaffold. The classes associated to Step 1 are grouped in `PySkelton.skeleton` and `PySkelton.graph` submodules, while `PySkelton.field` groups the classes needed in Step 2. The scaffold method is in `PySkelton.scaffolder` and the meshing method is in `PySkelton.mesher`. The visualization is handled by the classes in `PySkelton.visualization`.

[1] `https://gitlab.inria.fr/afuentes/pyskelton`

6.1.1 Skeletons

The skeletons in **PySkelton** are modeled with the classes `Segment`, `Arc`, and `G1Curve`, in the submodule `PySkelton.skeleton`. These classes inherit from the abstract class `Skeleton` that provides the common functionality (Figure 6.1). Every `Skeleton` (and derived) object represents a spatial curve $\Gamma([0,l]) \subset \mathbb{R}^3$ and must provide a valid frame for it (see Section 5.2.2). The most important methods in `Skeleton` are of the form `get_<elem>_at(t)` where $t \in [0,l]$ and `<elem>` can be any of `point`, `tangent`, `normal`, `binormal`, or `frame`. The main parameters for the constructors of skeleton classes are described below.

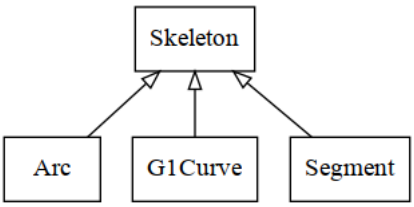


Figure 6.1: UML diagram of `PySkelton.skeleton` submodule.

`Segment`: Base point P, director vector $\mathbf{v}$, length l, and normal vector $\mathbf{n}$. This defines the line segment $P + s\,\mathbf{v}$ with $s \in [0,l]$. The normal vector is necessary to unambiguously define the frame of the line segment.

`Arc`: Center C, base vectors $\mathbf{u}$ and $\mathbf{v}$, radius r and spanning angle ϕ. This defines the arc of circle $C + r\,\mathbf{u}\cos\frac{s}{r} + r\,\mathbf{v}\sin\frac{s}{r}$ with $s \in [0, r\phi]$. The frame for this curve is given by the Frenet frame of the arc.

`G1Curve`: `skels`, collection of segments and arcs that form the circular spline. The frame at each point is given by the corresponding skeleton piece in the spline with the extra rotation automatically computed to make the frame continuous (see Section 5.2.2).

The scaffolding algorithm needs a graph as input, hence the user needs to build a `Graph` object representing the connectivity of the skeleton. For arcs of circle or $\mathscr{G}^1$ curves the skeleton is represented by a tangential polyline (see Section 5.4). The tangential polyline is specified directly by the user. The two method that incrementally build a graph are:

- `Graph.add_node(P)` it adds a node $P \in \mathbb{R}^3$ to the graph returning its index in the list of nodes.
- `Graph.add_edge(i,j)` it adds an edge between nodes with index i and j.

When adding nodes, if the node is already present then the index is directly returned. Graphs can be saved to a file with `Graph.save_to_graph_file`, and loaded with `Graph.read_from_graph_file`. The content of a graph file is a list of node coordinates preceded by the keyword *nodes* and a list of edges preceded by the keyword *edges* as shown below.

```
nodes
x0 y0 z0
...
xn yn zn
edges
i0 j0
...
im jm
```

Extra numerical information can be associated to nodes of the graph by adding a line with a tag (any word without digits) followed by $n+1$ lines, each with a numerical quantity, that are associated to the $n+1$ nodes of the graph. In this way one can add, for example, radii information to the graph. The extra information is stored in the dictionary `Graph.data` that maps the tags to the list of numerical values.

6.1.2 Fields

Fields are handled by the base class `Field`, and derived classes `SegmentField`, `ArcField` and `G1Field`. The class `MultiField` is in charge of taking the list of fields of the whole skeleton and computing the actual convolution field of the surface. The computation is done by summing all the values of the fields for each skeleton piece multiplied by its corresponding coefficient (by default 1) as described in Equation (5.3.2). These classes are part of the submodule `PySkelton.field` (the UML class diagram is shown in Figure 6.1).

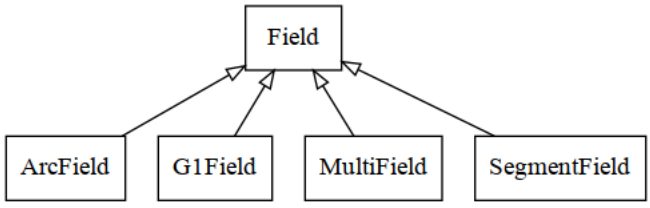


Figure 6.2: UML diagram of `PySkelton.field` submodule.

We describe next the two most important methods of the class `Field`.

`Field.eval` Parameter: $X \in \mathbb{R}^3$. Returns the value of the field at X. The evaluation depends on the specific type of field. A `Multifield` and `G1Field` return the sum of the values in their list of fields. `SegmentField` and `ArcField` call optimized numerical integration code that is implemented in `C`.

`Field.shoot_ray` Parameters: $Q \in \mathbb{R}^3$, $\mathbf{m}$ unit vector, c the level value of the surface. Returns a point $Q' = Q + t_0\mathbf{m} \in \mathbb{R}^2$ such that the field evaluated at Q' equals c.

In Section 6.3 we discuss implementation details of these two methods.

The constructors of `SegmentField` and `ArcField` receives the parameters: `skel` skeleton object, $a = (a_0, a_1)$ the tangential radii, $b = (b_0, b_1)$ normal radii, $c = (c_0, c_1)$ binormal radii, and $\theta = (\theta_0, \theta_1)$ rotation angles of the frame with respect to the Frenet frame (see Section 5.2.2). The pairs a, b, c, and θ define the values at the endpoints for the radii and rotation information. Along the skeleton the radii are interpolated as described in Equation (5.3.10), while the rotation angle is interpolated linearly. The utility function `PySkelton.make_field` receives the parameters for a field and construct the adequate `Field` instance (according with the instance skeleton supplied as parameter).

6.1.3 Scaffolds

The scaffolding algorithm is implemented in the class `Scaffolder`. This class receives as parameters a graph object representing the skeleton. The actual computation of the scaffold is done by the method `Scaffolder.compute_scaffold` and its controlled by several parameters that can be set on the `Scaffolder` object. We list below the main methods to setup the scaffolding process.

`Scaffolder.min_subdivs` defines the minimum number of points on each cell, default: 4.

`Scaffolder.long_arc_angle` sets the maximum length for arcs on the boundary of cells, default: 0.9π.

`Scaffolder.radii` list of radii associated to each node of the skeleton graph. The i-th radius define the size of the sphere at the i-th node of the graph in the final scaffold.

`Scaffolder.regular` sets whether the final scaffold should be regular or not, default: `False`.

`Scaffolder.symmetric` sets whether the final scaffold should respect the symmetries of the skeleton, default: `False`.

`Scaffolder.symmetries` list of permutations in terms of node indices. Each permutation represents a skeleton symmetry where index j at position i means that the i-th node of the skeleton graph is mapped to the j-th node.

There are utility methods in the class `Scaffolder` that helps in the interaction with the parameters described above. Their names are self explanatory.

6.1.4 Meshing

The class `Mesher` is the most complex class in **PySkelton**. It is in charge of computing a mesh for the anisotropic convolution surface. This class implements the scaffold-based meshing technique described in Section 5.4. The parameters of the class are:

- `scaff` a `Scaffolder` instance representing the skeleton.
- `field` a `MultiField` instance representing the convolution field.
- `pieces` a list of pairs `(skeleton,list_of_indices)` that maps the skeletons of the model (instances of `Skeleton`) to the corresponding nodes of the skeleton graph (given as a list of indices).

The `Mesher` instance constructs a mesh by reusing the scaffold computed by `scaff`. It uses `pieces` to map information about links and cells to the skeleton pieces. That information is used in the projection of a scaffold based mesh using the convolution field given by `field`. The projection is done by using the method `Field.shoot_ray` provided by the `field` parameter.

`Mesher` instances have several parameters that control the creation of the mesh. We describe them below.

`Mesher.quads_num` sets the number of quads taken along each skeleton piece.

`Mesher.max_quads_size` sets the maximal skeleton length between the projection points of the extremities of quads along the skeleton. It overrides `Mesher.quads_num` when a positive value is provided.

`Mesher.cap_quads_num` sets the number of quads taken at dangling nodes in order to get a closed mesh at tips.

`Mesher.level_set` set the level value defining the convolution surface.

The construction of the mesh is done by the method `Mesher.compute`.

6.1.5 Visualization

The abstract class `Visualization` models the main components needed for a 3D visualization of either the scaffold or the surface mesh. Subclasses should implement the mapping from the components of the visualization to a specific rendering *backend*. Currently we have implemented a visualization that uses **Axl**[2] software as backend: `VisualizationAxel`. Below we describe the main methods of `Visualization`.

`Visualization.add_points` Parameter: `points` list of points. Adds a set of points to the visualization

`Visualization.add_polyline` Parameters: `points` list of points of the polyline, each point is assumed to be connected with the next point of the list with a straight segment.

`Visualization.add_mesh` Parameters: `mesh_points` list of vertices of the mesh, `mesh_facets` list of faces of the mesh. Each face is given by a list of vertex indices in `mesh_points`.

The methods described above have some common parameters: `name` a name for the component (set of points, polyline, or mesh), `color` color used for the visualization of the component.

`Visualization.show` Runs the rendering backend and shows the components.

`Visualization.save` Parameter: `fname` name of a file. Saves the visualization to a file in the backed specific format.

With the utility function `PySkelton.get_visualization` a default visualization instance is returned.

A `Mesher` or `Scaffolder` instance can automatically add components to a visualization with the methods `Mesher.draw` or `Scaffolder.draw` to a visualization passed in as parameter. In the case of a `Scaffolder` instance we can get a hexahedral representation by calling `Scaffolder.draw_hex`.

6.2 Examples

We provide some examples that illustrates the main components of **PySkelton**.

[2] `http://axl.inria.fr/`

6.2.1 Scaffold example

The code below computes a scaffold as shown in Figure 6.3. Note that we used the helper method `Scaffold.read_symmetries` to read the list of symmetries from a file. This function interprets each line of the file as a symmetry, it expects a comma separated list of node indices representing the symmetry permutation. We also added radii information to the graph file which is read and stored in `Graph.data`.

```
import PySkelton as pk
#create a graph
g = pk.Graph()
#load from a file
g.read_from_graph_file("Dragon.graph")
#init the scaffolder
scaff = pk.Scaffolder(g)
#read symmetries from a file
scaff.read_symmetries("Dragon.sym")
#set max length of arcs in cells
scaff.long_arc_angle = pk.pi/2
#set raddi of joints (read from the file Dragon.graph)
scaff.set_radii(g.data["radii"])
#perform the scaffold computation
scaff.compute_scaffold()
#init visualization
vis = pk.get_visualization()
#draw the scaffold
scaff.draw(vis)
#show the rendered
vis.show()
```

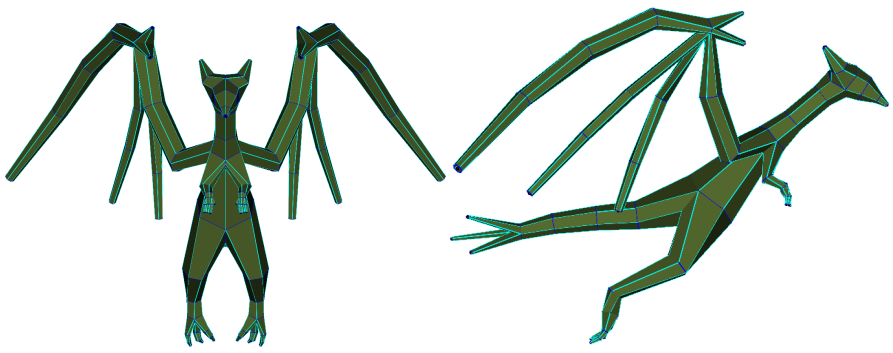


Figure 6.3: Scaffold representing a dragon computed with **PySkelton**.

6.2.2 Anisotropic convolution mesh example

To compute a mesh we need to setup all the parameters and compute first a scaffold, the code belows illustrates this process. We show the output in Figure 6.4. Note that we used the helper function `PySkelton.make_field` to build a `SegmentField`.

```
import PySkelton as pk

#create a graph of the skeleton
g = pk.Graph()
g.add_node([0.0,0.0,0.0])
g.add_node([3.0,0.0,-4.0])
g.add_edge(0,1)

#normal of the segment
n = [0.0,1.0,0.0]
#extremities of the segment
A,B = g.nodes[0],g.nodes[1]
#create segment skeleton
seg = pk.Segment.make_segment(A,B,n)
#define associated graph pieces
pieces = [(seg,[0,1])]

#create a SegmentField for the skeleton
field = pk.make_field(seg,a=[0.16,1.0],c=[0.16,0.16],\
b=[0.16,1.05],th=[0.0,3*pk.pi])

#Create and setup the scaffolder
scaff = pk.Scaffolder(g)
scaff.min_cell_quads = 40
scaff.compute_scaffold()

#create the mesher
mesher = pk.Mesher(scaff,field,pieces)
#set max quad size
mesher.max_quads_size = 0.1
#set number of quads at caps (tips of the surface)
mesher.cap_quads_num = 10
#perform computation of the mesh
mesher.compute()
```

```
vis = pk.get_visualization()
#draw the mesh
mesher.draw(vis)
vis.show()
```

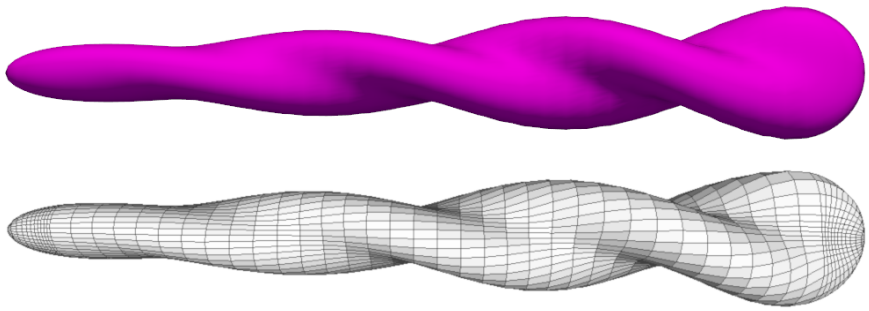


Figure 6.4: Mesh of an anisotropic convolution surface computed with **PySkelton**.

6.3 Implementation

In the implementation of **PySkelton** we addressed several issues. Not only the optimality of the numerical integration was necessary but also the the ray shooting process presented some challenges.

6.3.1 Integration

Numerical integration in **PySkelton** was implemented with GSL/QUADPACK. For the evaluation of `SegmentField` and `ArcField` we used the function `gsl_integration_qag` that performs and adaptive integration with the 61 points Gauss-Kronrod quadrature rule. The absolute and relative error are set to 10^{-8}. The adaptive workspace (maximal number of split segments) was set to 100. We already discussed this in Section 5.5. Here we add that in order to optimize more the field evaluation we perform first a distance computation between the skeletal curve (line segment or arc of circle) to discard the computation of points that are too far away, for which we can directly return a zero value for the field. This computation is done taking into account the maximal radii of the ellipsoid interpolation along the skeletal curve. We also discussed in Section 5.5 how considering the closest point in the skeletal curve as an integrand extreme improves the numerical value of the field. Alternatively we can use the function `gsl_integration_qagp` that takes as parameter a list of *singular* points in the interval of integration. The integration around a singularity uses more quadrature points in order to better control the error.

6.3.2 Ray shooting

To find the intersection of a ray starting at Q in the direction $\mathbf{m}$ (unit vector) we look for solutions of $f(t) = 0$ with f defined as

$$f(t) = C_{\Sigma}^{K}(Q + t\,\mathbf{m}) - c$$

where c is the level set defining the convolution surface.

We implemented the ray shooting process using Brent's method [Brent, 2002] for root finding without derivatives. This method is guaranteed to find a root in any interval $(a,b) \subset \mathbb{R}$ such that $f(a)$ and $f(b)$ have different signs. We are looking for the minimal value of $t > 0$ such that $f(t) = 0$ (this will be the first intersection point of the ray with the surface). We know that the convolution field, in general, decreases when going away from the skeleton, therefore we can find a value $b_0 > 0$ such that $f(0) > 0$ and $f(b_0) < 0$. In our implementation we take increasing values of b_0, starting at the maximal radii of the metric matrix, until $f(b_0) < 0$. We then apply the following process:

1. For each interval $(0, b_i)$ $(i = 0, 1, 2, \dots)$ we apply Brent's method to get a root $t_i \in (0, b_i)$ such that $f(t_i) = 0$.
2. Let $\mathscr{T}_i = \left\{ \frac{10-j}{10} t_i : j \in \{1, \dots, 9\} \text{ and } f\left(\frac{10-j}{10} t_i\right) < 0 \right\}$.
3. If $\mathscr{T}_i \neq \emptyset$ we take $t_{i+1} = \min \mathscr{T}_i$, then go to Step 1.
4. If $\mathscr{T}_i = \emptyset$ we return the root t_i.

This process guarantees that we get an intersection point. It also performs a reasonably good scan for discarding roots that are not the first intersection by sampling, in 10 subintervals, the interval $(0, \hat{t})$ where $\hat{t}$ is the latest found root (Step 2).

At first sight it appears that a Newton type method could be used in the ray shooting process but, in fact, there are some limitations. The convolution field behavior along a ray is sufficiently complex that it is not clear how to guarantee the preconditions for a correct Newton method (the derivate may vary in complex ways). In the literature [Kalra and Barr, 1989, Mitchell, 1990] ray intersection with general implicit surfaces (note that we do not have even closed form formulas) is usually done with a sampling approach. Guaranteeing that the ray is *hitting* the correct part of the surface is also a hard. Sampling is not enough to decide whether we have found the right root. One possibility is to reuse the neighboring information in a mesh projection to decide whether the intersection is correct or not: if a ray intersection is substantially further than its neighbors we increase the sampling to detect extra intersections. Another way to discard false-positives is to check the gradient, i.e. the normal to the surface, at the current root: if the ray direction $\mathbf{m}$ does not point outward as compared to the gradient then this is not the right intersection.

6.4 Conclusions

In this chapter we presented **PySkelton**, an academic-friendly Python package with our scaffolding algorithms and anisotropic convolution surfaces. We provided examples and documentation for future use. We also described some of the implementation details. **PySkelton** is currently being developed into a more optimal solution implemented in `C++`. The current Python package serves as a testing ground for new ideas and for quickly setting up prototypes. The `C++` will clone the functionality in **PySkelton** with the goal of developing plugins for established softwares.

Conclusions

In this thesis we addressed two main topics.

We first proposed a scaffolding method based on spherical Voronoi diagrams that uses integer linear programming. The main advantages of our method is that it works for skeletons of any topology, including skeletons with cycles. Our method does not add extra quads at joints, each quad corresponds to only one line segment. We can construct scaffold meshes that respect any subgroup (in the algebraic sense) of symmetries of the underlying skeletons. This includes reflection and rotation symmetries. The group of symmetries to be respected by the scaffold are chosen by the user from the symmetries of the skeleton. Another type of scaffolds that we can construct are regular ones. That is scaffolds with the same number of patches around each line segment. Symmetric regular scaffolds can be constructed as well. We presented formal proofs for the construction of symmetric and/or regular scaffolds. The integer linear programming model guarantees that the quad mesh obtained with our method are optimal: they have a minimal number of quads.

As further topics on scaffolding we discussed volumetric mesh generation and control over the shape around joints.

We presented a method to generate a hexahedral mesh for the volume enclosed by a scaffold. With this method the singular edges of the volumetric mesh are kept close to the skeleton. More research is needed on the hexahedral subdivision in order to study the geometric properties of the final hexahedra in the volumetric mesh.

Finer control over the shape of the scaffold can be achieved by using spherical Laguerre diagrams instead of Voronoi diagrams. We presented the geometrical ideas and algorithm variants that implement scaffolding with spherical Laguerre diagrams. There is need for more research. Although some ideas of the proof for the Voronoi setting can be reused, it is unclear how to guarantee feasibility. Another research path is on how to precisely set the weights governing the shape of the Laguerre regions.

As a second topic on skeleton-based geometric modeling, we introduced anisotropy to convolution surfaces by incorporating non-Euclidean metrics in the convolution field definition. This new technique provides ellipsoidal normal sections and good radii control in normal and tangential directions. Anisotropic convolution is also scale invariant, hence

it keeps the achievements of SCALIS – the state of the art in convolution surfaces. Our technique is simple and intuitive to use. It needs only eight parameters from the user along each $\mathscr{G}^1$ skeletal curve to be able to construct complex shapes around the skeleton. The new modeling capabilities allows to have non convex normal sections by adding the same skeletal curve with two metrics in the skeleton. We can mimic 2D skeletons, thus effectively decreasing the need of them.

Furthermore with the new capabilities we can use anisotropic convolution surfaces around 1D skeletons for shape approximation. Although promising, this new application needs further research on the optimization method for the defining parameters of the surfaces, as well as on the robustness of the process.

We also introduced Creative Telescoping as a way to obtain closed form formulas in convolution fields. We gave a general approach to obtain recurrence formulas for kernel families. We illustrated the approach with the power inverse kernel family. The resulting formulas may have many terms for high degree kernels. This pushed the symbolic approach to its limits. We thus opted for high performance numerical integration algorithms in our extension of convolution surfaces.

As an application of scaffolding we developed a skeleton-driven meshing technique for constructing quad-dominant meshes for anisotropic convolution surfaces. The method is based on a modification of the scaffold construction that builds a scaffold for circular splines. The quad-dominant mesh provided by the scaffold is then projected onto the surface. The final mesh has good edge-flow. The projection step is suitable to improvements, in particular GPU parallelization. The mesh projection based on ray shooting has some limitations when the anisotropy around joints is highly varying. The use of the spherical Laguerre diagrams variant for scaffolding may help in some cases. For a general setting we believe that an evolving mesh projection method (where the mesh is projected as a whole and taking into account the connectivity of the vertices) would be more robust.

All the methods and algorithms we presented in this work were implemented in a Python library. We described the general design of the library and gave some examples of use. In the short term we also envisage to make plugins for the software IceSL[3] and Axel[4] with the `C++` version of **PySkelton**.

In this thesis we focused on 1D skeletons. Both scaffolding and anisotropic convolution can be generalized to 2D skeletons. For purely 2D skeletons a scaffold can be obtained by offset [Angelidis and Cani, 2002]. A 2D extension to anisotropic convolution would require a well crafted cross field [Bunin, 2008, Panozzo et al., 2014] to support the metric matrix. Whether for scaffolding or for designing the cross field, a key challenge will be the interface between elements of different dimensions.

[3] `https://icesl.loria.fr/`
[4] `http://axel.inria.fr/`

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 675789.

Bibliography

[Angelidis and Cani, 2002] Angelidis, A. and Cani, M.-P. (2002). Adaptive implicit modeling using subdivision curves and surfaces as skeletons. *Proceedings of the seventh ACM symposium on Solid modeling and applications - SMA '02*, page 45.

[Augenbaum and Peskin, 1985] Augenbaum, J. M. and Peskin, C. S. (1985). On the construction of the Voronoi mesh on a sphere. *Journal of Computational Physics*, 59(2):177–192.

[Aurenhammer, 1991] Aurenhammer, F. (1991). Voronoi Diagrams — A Survey of a Fundamental Data Structure. *ACM Computing Surveys*, 23(3):345–405.

[Axler, 2015] Axler, S. (2015). *Linear Algebra Done Right*. Undergraduate Texts in Mathematics. Springer International Publishing.

[Bærentzen et al., 2014] Bærentzen, J. A., Abdrashitov, R., and Singh, K. (2014). Interactive shape modeling using a skeleton-mesh co-representation. *ACM Trans. Graph.*, 33(4):1–10.

[Bærentzen et al., 2012] Bærentzen, J. A., Misztal, M. K., and Wełnicka, K. (2012). Converting skeletal structures to quad dominant meshes. *Computers and Graphics*, 36(5):555–561.

[Barber et al., 1996] Barber, C. B., Dobkin, D. P., and Huhdanpaa, H. (1996). The Quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483.

[Barbieri et al., 2016] Barbieri, S., Meloni, P., Usai, F., Spano, L. D., and Scateni, R. (2016). An interactive editor for curve-skeletons: SkeletonLab. *Computers & Graphics*, 60:23–33.

[Bernhardt et al., 2008] Bernhardt, A., Pihuit, A., Cani, M.-P., and Barthe, L. (2008). Matisse: Painting 2D regions for Modeling Free-Form Shapes. In Alvarado, C. and Cani, M.-P., editors, *Eurographics Workshop on Sketch-Based Interfaces and Modeling*. The Eurographics Association.

[Bertails et al., 2006] Bertails, F., Audoly, B., Cani, M.-P., Querleux, B., Leroy, F., and Lévêque, J.-L. (2006). Super-helices for predicting the dynamics of natural hair. *ACM Transactions on Graphics*, 25(3):1180.

[Bishop, 1975] Bishop, R. L. (1975). There is More than One Way to Frame a Curve. *The American Mathematical Monthly*, 82(3):246–251.

[Blidia et al., 2017] Blidia, A., Mourrain, B., and Villamizar, N. (2017). G1-smooth splines on quad meshes with 4-split macro-patch elements. *Computer Aided Geometric Design*, 52-53:106–125.

[Blinn, 1982] Blinn, J. F. (1982). A Generalization of Algebraic Surface Drawing. *ACM Transactions on Graphics*, 1(3):235–256.

[Bloomenthal and Shoemake, 1991] Bloomenthal, J. and Shoemake, K. (1991). Convolution surfaces. *ACM SIGGRAPH Computer Graphics*, 25(4):251–256.

[Bolton, 1975] Bolton, K. (1975). Biarc curves. *Computer-Aided Design*, 7(2):89–92.

[Bostan et al., 2013] Bostan, A., Chen, S., Chyzak, F., Li, Z., and Xin, G. (2013). Hermite reduction and creative telescoping for hyperexponential functions. In *Proceedings of the 38th international symposium on International symposium on symbolic and algebraic computation - ISSAC '13*, number i, page 77, New York, New York, USA. ACM Press.

[Bostan et al., 2016] Bostan, A., Dumont, L., and Salvy, B. (2016). Efficient Algorithms for Mixed Creative Telscoping. In *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation*, ISSAC '16, pages 127–134, New York, NY, USA. ACM.

[Botsch et al., 2010] Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., and Levy, B. (2010). *Polygon Mesh Processing*. A K Peters.

[Brent, 2002] Brent, R. P. (2002). *Algorithms for Minimization Without Derivatives*. Dover Books on Mathematics. Dover Publications.

[Bronstein, 2005] Bronstein, M. (2005). *Symbolic Integration I*. Springer-Verlag.

[Brown, 1979] Brown, K. Q. (1979). *Geometric Transforms for Fast Geometric Algorithms*. PhD thesis, Carnegie-Mellon University.

[Bunin, 2008] Bunin, G. (2008). A continuum theory for unstructured mesh generation in two dimensions. *Computer Aided Geometric Design*, 25(1):14–40.

[Burden and Faires, 2011] Burden, R. L. and Faires, J. D. (2011). *Numerical Analysis*. Richard Stratton.

[Canezin et al., 2013] Canezin, F., Guennebaud, G., and Barthe, L. (2013). Adequate inner bound for geometric modeling with compact field functions. *Computers & Graphics*, 37(6):565–573.

[Cani and Hornus, 2001] Cani, M.-P. and Hornus, S. (2001). Subdivision-curve primitives: a new solution for interactive implicit modeling. In *Proceedings - International Conference on Shape Modeling and Applications, SMI 2001*, pages 82–88. IEEE Computer Society.

[Casti et al., 2019] Casti, S., Livesu, M., Mellado, N., Rumman, N. A., Scateni, R., Barthe, L., and Puppo, E. (2019). Skeleton based cage generation guided by harmonic fields. *Computers & Graphics*.

[Chen et al., 2009] Chen, D.-S., Batson, R. G., and Dang, Y. (2009). *Applied Integer Programming*. John Wiley & Sons, Inc., Hoboken, NJ, USA.

[Chen and Kauers, 2012] Chen, S. and Kauers, M. (2012). Trading order for degree in creative telescoping. *Journal of Symbolic Computation*, 47(8):968–995.

[Chen and Kauers, 2017] Chen, S. and Kauers, M. (2017). Some open problems related to creative telescoping. *Journal of Systems Science and Complexity*, 30(1):154–172.

[Chyzak and Salvy, 1998] Chyzak, F. and Salvy, B. (1998). Non-commutative Elimination in Ore Algebras Proves Multivariate Identities. *Journal of Symbolic Computation*, 26(2):187–227.

[Dillencourt and Smith, 1996] Dillencourt, M. B. and Smith, W. D. (1996). Graph-theoretical conditions for inscribability and Delaunay realizability. *Discrete Mathematics*, 161(1):63–77.

[do Carmo, 1976] do Carmo, M. P. (1976). *Differential geometry of curves and surfaces*. Prentice-Hall, New Jersey.

[Douglas and Peucker, 1973] Douglas, D. H. and Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122.

[Entem et al., 2015] Entem, E., Barthe, L., Cani, M.-P., Cordier, F., and van de Panne, M. (2015). Modeling 3d animals from a side-view sketch. *Computers & Graphics*, 46:221–230.

[Evans, 2015] Evans, L. C. (2015). *Measure Theory and Fine Properties of Functions, Revised Edition*. Chapman and Hall/CRC.

[Federer, 1996] Federer, H. (1996). *Geometric Measure Theory*. Springer Berlin Heidelberg.

[Flanders, 1973] Flanders, H. (1973). Differentiation under the integral sign. *American Mathematical Monthly*, 80(6):615–627.

[Fryazinov and Pasko, 2017] Fryazinov, O. and Pasko, A. (2017). Implicit variable-radius arc canal surfaces for solid modeling. *Computer-Aided Design and Applications*, 14(3):251–258.

[Fuentes Suárez and Hubert, 2017] Fuentes Suárez, A. J. and Hubert, E. (2017). Scaffolding skeletons using spherical Voronoi diagrams. *Electronic Notes in Discrete Mathematics*, 62:45–50.

[Fuentes Suárez and Hubert, 2018a] Fuentes Suárez, A. J. and Hubert, E. (2018a). Convolution surfaces with varying radius: Formulae for skeletons made of arcs of circles and line segments. In *Research in Shape Analysis: WiSH2, Sirince, Turkey*, AWM, pages 37–60. Springer.

[Fuentes Suárez and Hubert, 2018b] Fuentes Suárez, A. J. and Hubert, E. (2018b). Scaffolding skeletons using spherical Voronoi diagrams: Feasibility, regularity and symmetry. *Computer-Aided Design*, 102:83–93.

[Fuentes Suárez et al., 2019] Fuentes Suárez, A. J., Hubert, E., and Zanni, C. (2019). Anisotropic convolution surfaces. *Computers & Graphics*, 82:106–116.

[Galassi et al., 2017] Galassi, M., Davies, J., Theiler, J., Gough, B., Jungman, G., Alken, P., Booth, M., Rossi, F., and Ulerich, R. (2017). *GNU Scientific Library*. Number Release 2.4. GNU.

[Gomes et al., 2009] Gomes, A. J., Voiculescu, I., Jorge, J., Wyvill, B., and Galbraith, C. (2009). *Implicit Curves and Surfaces: Mathematics, Data Structures and Algorithms*. Springer London.

[Grima and Márquez, 2001] Grima, C. I. and Márquez, A. (2001). *Computational Geometry on Surfaces*. Springer Netherlands.

[Hart, 2002] Hart, G. W. (2002). Solid-segment sculptures. In *Mathematics and Art*, pages 17–27. Springer Berlin Heidelberg.

[Hart, 2008] Hart, G. W. (2008). Sculptural forms from hyperbolic tessellations. In *2008 IEEE International Conference on Shape Modeling and Applications*. IEEE.

[Hornus et al., 2003] Hornus, S., Angelidis, A., and Cani, M.-P. (2003). Implicit modeling using subdivision curves. *The Visual Computer*, 19(2):94–104.

[Hubert, 2012] Hubert, E. (2012). Convolution surfaces based on polygons for infinite and compact support kernels. *Graphical Models*, 74(1):1–13.

[Hubert and Cani, 2012] Hubert, E. and Cani, M.-P. (2012). Convolution surfaces based on polygonal curve skeletons. *Journal of Symbolic Computation*, 47(6):680–699.

[Ji et al., 2010] Ji, Z., Liu, L., and Wang, Y. (2010). B-Mesh: A modeling system for base meshes of 3D articulated shapes. *Computer Graphics Forum*, 29(7):2169–2177.

[Jin and Tai, 2002a] Jin, X. and Tai, C.-L. (2002a). Analytical methods for polynomial weighted convolution surfaces with various kernels. *Computers & Graphics*, 26(3):437–447.

[Jin and Tai, 2002b] Jin, X. and Tai, C.-L. (2002b). Convolution surfaces for arcs and quadratic curves with a varying kernel. *The Visual Computer*, 18(8):530–546.

[Jin et al., 2001] Jin, X., Tai, C.-L., Feng, J., and Peng, Q. (2001). Convolution surfaces for line skeletons with polynomial weight distributions. *Journal of Graphics Tools*, 6(3):17–28.

[Jin et al., 2008] Jin, X., Tai, C.-L., and Zhang, H. (2008). Implicit modeling from polygon soup using convolution. *The Visual Computer*, 25(3):279–288.

[Kalra and Barr, 1989] Kalra, D. and Barr, A. H. (1989). Guaranteed ray intersections with implicit surfaces.

[Karčiauskas and Peters, 2016] Karčiauskas, K. and Peters, J. (2016). Curvature continuous bi-4 constructions for scaffold- and sphere-like surfaces. *CAD Computer Aided Design*, 78:48–59.

[Karp, 2009] Karp, R. M. (2009). Reducibility among combinatorial problems. In *50 Years of Integer Programming 1958-2008*, pages 219–241. Springer Berlin Heidelberg.

[Koutschan, 2010] Koutschan, C. (2010). A fast approach to creative telescoping. In *Mathematics in Computer Science*.

[Koutschan, 2013] Koutschan, C. (2013). *Creative Telescoping for Holonomic Functions*, pages 171–194. Springer Vienna.

[Livesu et al., 2017] Livesu, M., Attene, M., Patané, G., and Spagnuolo, M. (2017). Explicit cylindrical maps for general tubular shapes. *Computer-Aided Design*, 90:27–36.

[Livesu et al., 2016] Livesu, M., Muntoni, A., Puppo, E., and Scateni, R. (2016). Skeleton-driven Adaptive Hexahedral Meshing of Tubular Shapes. *Computer Graphics Forum*, 35(7):237–246.

[Livesu and Scateni, 2013] Livesu, M. and Scateni, R. (2013). Extracting curve-skeletons from digital shapes using occluding contours. *Visual Computer*, 29(9):907–916.

[Lorensen and Cline, 1987] Lorensen, W. E. and Cline, H. E. (1987). Marching cubes: A high resolution 3d surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 21(4):163–169.

[Ma and Crawford, 2008] Ma, G. and Crawford, R. H. (2008). Topological consistency in skeletal modeling with convolution surfaces. In *Volume 3: 28th Computers and Information in Engineering Conference, Parts A and B*, pages 307–315. ASME.

[Makhorin, 2016] Makhorin, A. (2016). *GNU Linear Programming Kit - Reference Manual*. https://www.gnu.org/software/glpk/.

[McCormack and Sherstyuk, 1998] McCormack, J. and Sherstyuk, A. (1998). Creating and rendering convolution surfaces. *Computer Graphics Forum*, 17(2):113–120.

[Meek and Walton, 2008] Meek, D. and Walton, D. (2008). The family of biarcs that matches planar, two-point G^1 Hermite data. *Journal of Computational and Applied Mathematics*, 212(1):31–45.

[Milnor, 1963] Milnor, J. W. (1963). Morse Theory. page 153.

[Mitchell, 1990] Mitchell, D. (1990). Robust ray intersection with interval arithmetic. *Proceedings of Graphics Interface '90*.

[Na et al., 2002] Na, H.-S., Lee, C.-N., and Cheong, O. (2002). Voronoi diagrams on the sphere. *Computational Geometry*, 23(2):183–194.

[Notaris, 2016] Notaris, S. E. (2016). Gauss-Kronrod quadrature formulae - A survey of fifty years of research. *Electronic Transactions on Numerical Analysis*, 45:371–404.

[Nutbourne and Martin, 1988] Nutbourne, A. W. and Martin, R. R. (1988). *Differential geometry applied to curve and surface design*. John Wiley & Sons.

[Ore, 1933] Ore, O. (1933). Theory of Non-Commutative Polynomials. *The Annals of Mathematics*, 34(3):480.

[Panetta et al., 2017] Panetta, J., Rahimian, A., and Zorin, D. (2017). Worst-case stress relief for microstructures. *ACM Transactions on Graphics*, 36(4):1–16.

[Panetta et al., 2015] Panetta, J., Zhou, Q., Malomo, L., Pietroni, N., Cignoni, P., and Zorin, D. (2015). Elastic textures for additive fabrication. *ACM Transactions on Graphics*, 34(4):135:1–135:12.

[Panotopoulou et al., 2018] Panotopoulou, A., Ross, E., Welker, K., Hubert, E., and Morin, G. (2018). Scaffolding a skeleton. In *Research in Shape Analysis: WiSH2, Sirince, Turkey*, AWM, pages 17–35. Springer.

[Panozzo et al., 2014] Panozzo, D., Puppo, E., Tarini, M., and Sorkine-Hornung, O. (2014). Frame Fields: Anisotropic and Non-Orthogonal Cross Fields. *ACM Transactions on Graphics*, 33(4):1–11.

[Pasko et al., 1995] Pasko, A., Adzhiev, V., Sourin, A., and Savchenko, V. (1995). Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer*, 11(8):429–446.

[Peternell and Pottmann, 1997] Peternell, M. and Pottmann, H. (1997). Computing rational parametrizations of canal surfaces. *Journal of Symbolic Computation*, 23(2-3):255–266.

[Pham, 1992] Pham, B. (1992). Offset curves and surfaces: a brief survey. *Computer-Aided Design*, 24(4):223–229.

[Piessens et al., 1983] Piessens, R., de Doncker-Kapenga, E., Überhuber, C. W., and Kahaner, D. K. (1983). *Quadpack*. Springer Series in Computational Mathematics. Springer Berlin Heidelberg.

[Requicha, 1980] Requicha, A. G. (1980). Representations for rigid solids: Theory, methods, and systems. *ACM Computing Surveys*, 12(4):437–464.

[Ricci, 1973] Ricci, A. (1973). A constructive geometry for computer graphics. *The Computer Journal*, 16(2):157–160.

[Rivin, 1996] Rivin, I. (1996). A characterization of ideal polyhedra in hyperbolic 3-space. *Annals of Mathematics*, 143(1):51–70.

[Roussellet et al., 2018] Roussellet, V., Rumman, N. A., Canezin, F., Mellado, N., Kavan, L., and Barthe, L. (2018). Dynamic implicit muscles for character skinning. *Computers & Graphics*, 77:227–239.

[Sherstyuk, 1999a] Sherstyuk, A. (1999a). Interactive shape design with convolution surfaces. In *Proceedings Shape Modeling International 99. International Conference on Shape Modeling and Applications*, pages 56–65. IEEE.

[Sherstyuk, 1999b] Sherstyuk, A. (1999b). Kernel functions in convolution surfaces: a comparative analysis. *The Visual Computer*, 15(4):171–182.

[Song et al., 2009] Song, X., Aigner, M., Chen, F., and Jüttler, B. (2009). Circular spline fitting using an evolution process. *Journal of Computational and Applied Mathematics*, 231(1):423–433.

[Srinivasan et al., 2005] Srinivasan, V., Mandal, E., and Akleman, E. (2005). Solidifying Wireframes. In *Bridges: Mathematical Connections in Art, Music, and Science 2004*, pages 203–210, Banf, Alberta, Canada.

[Sugihara, 2002] Sugihara, K. (2002). Laguerre voronoi diagram on the sphere. *Journal for Geometry and Graphics*, 6(1):69–81.

[Tagliasacchi et al., 2012] Tagliasacchi, A., Alhashim, I., Olson, M., and Zhang, H. (2012). Mean Curvature Skeletons. *Computer Graphics Forum*, 31(5):1735–1744.

[Tagliasacchi et al., 2016] Tagliasacchi, A., Delame, T., Spagnuolo, M., Amenta, N., and Telea, A. (2016). 3D Skeletons: A State-of-the-Art Report. *Computer Graphics Forum*, 35(2):573–597.

[Tai et al., 2004] Tai, C.-L., Zhang, H., and Fong, J. C.-K. (2004). Prototype modeling from sketched silhouettes based on convolution surfaces. *Computer Graphics Forum*, 23(1):71–83.

[Trefethen, 2008] Trefethen, L. N. (2008). Is Gauss quadrature better than Clenshaw-Curtis? *SIAM Review*, 50(1):67–87.

[Usai et al., 2015] Usai, F., Livesu, M., Puppo, E., Tarini, M., and Scateni, R. (2015). Extraction of the Quad Layout of a Triangle Mesh Guided by Its Curve Skeleton. *ACM Transactions on Graphics*, 35(1):1–13.

[Wang and Joe, 1997] Wang, W. and Joe, B. (1997). Robust computation of the rotation minimizing frame for sweep surface modeling. *Computer-Aided Design*, 29(5):379–391.

[Wang et al., 2008] Wang, W., Jüttler, B., Zheng, D., and Liu, Y. (2008). Computation of rotation minimizing frames. *ACM Transactions on Graphics*, 27(1):1–18.

[Wenger, 2013] Wenger, R. (2013). *Isosurfaces*. A K Peters/CRC Press, New York.

[Wither et al., 2009] Wither, J., Boudon, F., Cani, M.-P., and Godin, C. (2009). Structure from silhouettes: a new paradigm for fast sketch-based design of trees. *Computer Graphics Forum*, 28(2):541–550.

[Wu and Liu, 2012] Wu, J. and Liu, L. (2012). Generating quad mesh of 3d articulated shape for sculpting modeling. *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, 6(3):354–365.

[Wyvill et al., 1999] Wyvill, B., Guy, A., and Galin, E. (1999). Extending the CSG tree. Warping, blending and boolean operations in an implicit surface modeling system. *Computer Graphics Forum*, 18(2):149–158.

[Wyvill et al., 1986] Wyvill, G., McPheeters, C., and Wyvill, B. (1986). Data structure for soft objects. *The Visual Computer*, 2(4):227–234.

[Yan et al., 2016] Yan, Y., Sykes, K., Chambers, E., Letscher, D., and Ju, T. (2016). Erosion thickness on medial axes of 3D shapes. *ACM Transactions on Graphics*, 35(4):1–12.

[Yao et al., 2009] Yao, C.-Y., Chu, H.-K., Ju, T., and Lee, T.-Y. (2009). Compatible quadrangulation by sketching. *Computer Animation and Virtual Worlds*, 20(2-3):101–109.

[Zanni, 2013] Zanni, C. (2013). *Skeleton-based Implicit Modeling & Applications*. PhD thesis, Université de Grenoble.

[Zanni et al., 2013] Zanni, C., Bernhardt, A., Quiblier, M., and Cani, M.-P. (2013). SCALe-invariant integral surfaces. *Computer Graphics Forum*, 32(8):219–232.

[Zanni et al., 2011] Zanni, C., Hubert, E., and Cani, M.-P. (2011). Warp-based helical implicit primitives. *Computers & Graphics*, 35(3):517–523.

[Zhu et al., 2011] Zhu, X., Jin, X., Liu, S., and Zhao, H. (2011). Analytical solutions for sketch-based convolution surface modeling on the GPU. *The Visual Computer*, 28(11):1115–1125.

[Zhu et al., 2015a] Zhu, X., Jin, X., and You, L. (2015a). Analytical solutions for tree-like structure modelling using subdivision surfaces. *Computer Animation and Virtual Worlds*, 26(1):29–42.

[Zhu et al., 2015b] Zhu, X., Jin, X., and You, L. (2015b). High-quality tree structures modelling using local convolution surface approximation. *The Visual Computer*, 31(1):69–82.

[Zhu et al., 2017] Zhu, X., Song, L., You, L., Zhu, M., Wang, X., and Jin, X. (2017). Brush2Model: Convolution surface-based brushes for 3D modelling in head-mounted display-based virtual environments. *Computer Animation and Virtual Worlds*, 28(3-4).