Instituto Nacional de Matemática Pura e Aplicada

Geodesic-based Modeling on Manifold Triangulations

author: Dimas Martínez Morera

advisors: Paulo Cezar Carvalho
Luiz Velho

August, 2006

Abstract

We present a new algorithm to compute a geodesic path over a triangulated surface. Based on Sethian's Fast Marching Method and Polthier's Straightest Geodesics theory, we are able to generate an iterative process to obtain a good discrete geodesic approximation. It can handle both convex and non-convex surfaces.

We define a new class of curves, called *geodesic Bézier curves*, that are suitable for modeling on manifold triangulations. As a natural generalization of Bézier curves, the new curves are as smooth as possible. We discuss the construction of C^0 and C^1 piecewise Bézier splines. We also describe how to perform editing operations, such as trimming, using these curves. Special care is taken to achieve interactive rates for modeling tasks.

After giving an appropriated definition of convex sets on triangulations, we use it to study the convergence of the geodesic algorithm, as well as the convex hull property of geodesic Bézier curves. We prove some results concerning convex sets.

Keywords: Discrete Geodesic, Geodesic Bézier Curve, Manifold Triangulation, de Casteljau Algorithm, Discrete Differential Geometry, Spline Curves, Free-Form Design, Convex Sets on Triangulations.

Resumo

Neste trabalho apresentamos um novo algoritmo para calcular geodésicas em triangulações. Baseado no "Fast Marching Method" desenvolvido por Sethian e na teoria de "Straightest Geodesics" de Polthier, geramos um processo iterativo que permite obter uma boa aproximação da geodésica discreta. Nosso algoritmo funciona tanto em superfícies convexas quanto não convexas.

Definimos uma nova classe de curvas, chamadas de *curvas de Bézier geodésicas*, que são adequadas para modelagem em triangulações. Sendo uma generalização das curvas de Bézier planas, as novas curvas são tão suaves quanto possìvel. Discutimos a construção de splines de Bézier C^0 e C^1. Também descrevemos um modo de realizar operações de edição, como "trimming", usando estas curvas.

Damos uma definição alternativa de conjuntos convexos em triangulações, a qual é mais adequada ao estudo da convergência do algoritmo para calcular geodésicas. Esta definição também permite provar a propriedade do fecho convexo das curvas de Bézier geodésicas. Além disso provamos algumas propriedades relativas aos conjuntos convexos em triangulações.

Keywords: Geodésica Discreta, Curva de Bézier Geodésica, Algoritmo de de Casteljau, Conjuntos Convexos em Triangulações, Geometria Diferencial Discreta.

Acknowledgements

I would like to express my gratitude to those who contributed to the development of this thesis.

I am grateful to professors Paulo Cezar Carvalho and Luiz Velho by their guidance and encouragement during this research work. The suggestions of professor Luiz Henrique de Figueiredo were very helpful, especially in the subject of chapter 3. Discussions with professor Adán Corcho about convex sets improved the exposition of the results of chapter 4, as well as their mathematical formalism.

Without the financial support from CNPQ and UMALCA, it would not be possible the development of this thesis. So I am very grateful to these institutions.

The wonderful research environment of IMPA and the excellence of its staff contributed much to bring this work to a good end. In particular, the people from the VISGRAF Laboratory were of great importance. The implementation of the algorithms described here was possible thanks to their valuable help, particularly Ives Macedo and Adelailson Peixoto. The help of Margareth Catoia Varela in discovering the insights of FLTK was fundamental. Every color in the figures of this work was selected by Geisa Faustino. I wish also to express my gratitude to other friends from IMPA: Freddy, Javier, Sofía, Juan Carlos, Zé, Lourena, Perfilino, Nair, Ary, Esdras, Emilio, Nelma, Sergio, Daniel, William, Aryana, Francisco, Duilio, LG, André.

Even being geographically distant, the colleagues of the Geometry Group of ICIMAF were very important during these years. We had some useful discussions about this work and I had the opportunity of present parts of it in the group's seminary. ICIMAF stands for Instituto de Cibernética, Matemática y Física; it is located in Havana, Cuba.

Finally, I wish to express my enormous gratitude to my wife Alessandra by her fondness, love, encouragement and especially by her patience. I wish also thank my parents who guided me through the live and encourage me to study, showing me its importance since I was still a child.

Contents

1 Introduction

Discrete surfaces, especially triangulations, are present in many applications ranging from simple 3D objects to sophisticated mechanical engines or complex scanning models. Consequently, a wide range of modeling techniques have been developed [3, 4, 6, 20, 22, 36] to create or modify them. Most of these techniques rely on the use of parameterizations, carrying the difficulties associated to their use.

A great amount of effort has been devoted to obtaining good parameterizations for discrete surfaces. A very good reference for this subject can be found on the tutorials of Floater and Hormann [12, 13] and the references therein. However, the selection of the parameterization appropriated for certain application depends both on the surface and the application itself.

In this thesis we develop an algorithm to compute geodesic curves on triangulations. Since it does not depend on a parameterization, any modeling operation depending on geodesics will also be parameterization independent. That property permitted us to develop a simple geodesic-based toolbox to do modeling operations directly on the mesh without facing the problems associated to parameterizations. The definition and study of geodesic Bézier curves, as well as the construction of C^0 and C^1 splines, help us to design complicated curves on the surface, or to determine and select a region that can be trimmed, have a texture mapped to it, or even be painted with a color different from the rest of the surface. Those basic operations are, several times, the starting point for more sophisticated applications.

1.1 Notations and Preliminary Definitions

1.1.1 Discrete Surfaces

We restrict this work to the study of curves defined on manifold triangulations. Thus, we will use the following definition of discrete surfaces.

Definition 1.1 *A* discrete surface $\mathcal{S}$ *is a finite set* F *of (triangular) faces such that:*

1. *Any point in* $\mathcal{S}$ *lies in at least one triangle in* F*.*
2. *The intersection of two different triangles in* F *is either empty, or consists of a common vertex, or of a common edge.*
3. *Each point in* $\mathcal{S}$ *has a neighborhood which is either homeomorphic to a disc, or homeomorphic to a semi-disc.*

Let P be a vertex of the discrete surface $\mathcal{S}$, and θ be the sum of all its incident angles. The discrete Gaussian curvature of $\mathcal{S}$ at P is given by $\kappa(P) = 2\pi - \theta$ [26]. The vertices of discrete surfaces are classified according to Gaussian curvature:

Definition 1.2 *A mesh vertex is classified according to the sum θ of its incident angles as:*

1. *Euclidean if* $2\pi - \theta = 0$,
2. *Spherical if* $2\pi - \theta > 0$, *or*
3. *Hyperbolic if* $2\pi - \theta < 0$.

1.1.2 Curve Notations

We denote by a greek letter (α, β, γ, ...) a curve over a smooth surface, and use capital greek letters ($\mathcal{A}$, $\mathcal{B}$, Γ, ...) to denote curves over discrete surfaces.

A polygonal line Γ on a triangle mesh $\mathcal{S}$ is defined as a sequence of nodes $\{P_0, P_1, \ldots, P_n\} \subset \mathcal{S}$ such that every line segment $\overline{P_i P_{i+1}}$ is also contained in $\mathcal{S}$. We refer to polygonal vertices as nodes, in order to differentiate them from mesh vertices.

1.1.3 Length Functional

The length functional $L(\gamma) = \text{length}(\gamma)$ defined on the set of curves over a smooth surface $\mathcal{G}$ can be extended to $\mathcal{S}$ as:

$$L(\Gamma) = \sum_{f \in F} L(\Gamma_{|f}),$$

where $L(\Gamma_{|f})$ is measured according to the Euclidean metric in face f.

1.1.4 Continuity

Curves defined over a triangulation $\mathcal{S}$ cannot be smooth, the only exception being when they are completely defined on a planar part of $\mathcal{S}$, which is usually not the case. However, as continuity is a local property we can analyze the intrinsic behavior of a curve $\mathcal{C}$ on $\mathcal{S}$ by looking at the intersection of the neighborhoods of each curve point with $\mathcal{S}$.

We define C^k continuity at points having a neighborhood isometric to a plane as the usual C^k continuity in the unfolding of that neighborhood. In particular, this applies to points lying in the interior of mesh edges and faces. In the case of mesh vertices, continuity is not well defined.

1.2 Overview

Chapter 2 describes an iterative algorithm to compute geodesic paths on triangulations. Two different iteration strategies to reach the geodesic curve are presented. This algorithm is very suitable for the modeling tasks discussed later in this work.

Geodesic Bézier curves, which are suitable for modeling on manifold triangulations, are defined in chapter 3. Some results about their smoothness and their similarity with classical Bézier curves are proved. It is also presented a discussion on the construction of C^0 and C^1 piecewise Bézier splines, and it is described how to perform editing operations, such as trimming, using these curves. Special care is taken to achieve interactive rates for modeling tasks.

After giving an appropriated definition of convex sets on triangulations, in chapter 4, we use it to study the convergence of the geodesic algorithm, as well as the convex hull property of geodesic Bézier curves. Some results concerning convex sets are proved. Finally in section 5 we give concluding remarks and indicate potential further research.

2 Discrete Geodesic Curves

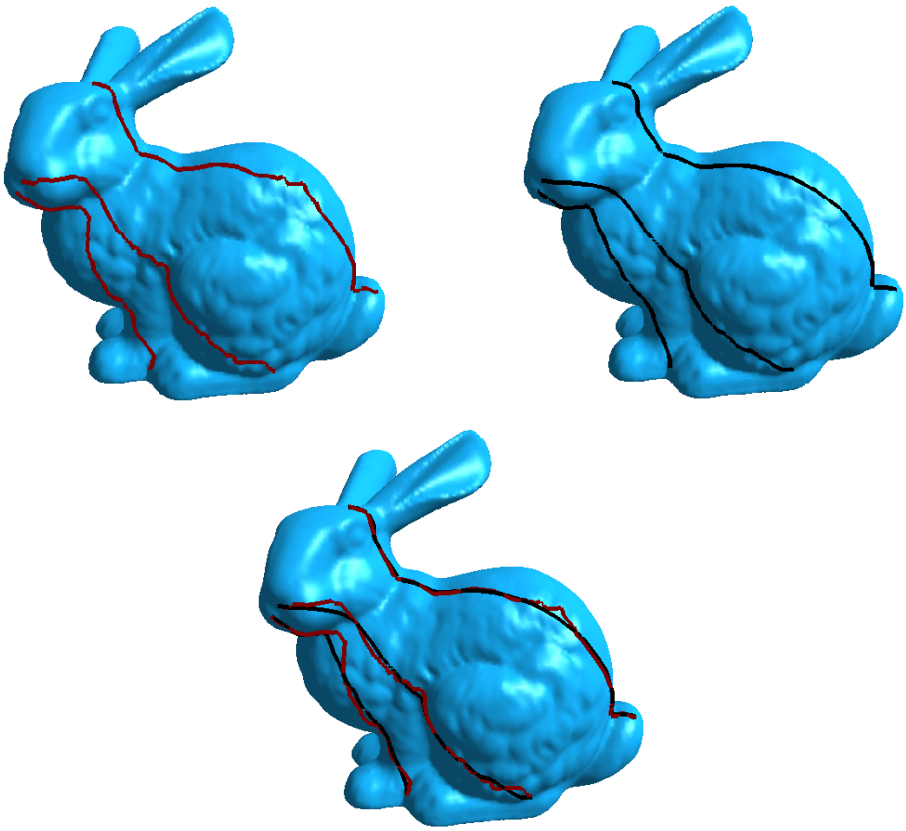


Figure 2.1: Geodesics over the Stanford bunny. The first approximations, the geodesics, and both.

Geodesic curves are useful in many areas of science and engineering, such as robot motion planning, terrain navigation, surface parameterization [12], remeshing [31] and front propagation over surfaces [27]. The increasing development of discrete surface models, as well as the use of smooth surfaces discretization to study their geometry, demanded the definition of Geodesic Curves for polyhedral surfaces [2, 26], and hence the study of efficient algorithms to compute them.

Such curves are called *Discrete Geodesics* and there exist some different definitions for them, mostly depending on the application in which they are used. Considering a geodesic as a shortest path between two points on the surface is perhaps the more widely used definition.

In this chapter we are concerned with the problem of computing a locally shortest geodesic joining two points over the surface. There are some slightly different formulations for this problem. The simplest version is the *single source shortest path problem*, in which one wishes to find a shortest path between a source point and any other point on the surface. Another, more complex, version of the problem asks for a subdivision on the surface such that a shortest path between any pair of points in the surface can be found quickly; this is known as the *all pairs shortest path problem*.

Most of the algorithms use front propagation or some other kind of Dijkstra's-like algorithm. In 1987 Mitchell, Mount and Papadimitriou [24] proposed the Continuous Dijkstra technique to build a data structure from which we can find a shortest path between the source and any point in time $O(k + \log m)$, where k is the number of faces crossed by the path and m is the number of mesh edges. This algorithm runs in $O(m^2 \log m)$ time and requires $O(m^2)$ space. In 1999 Kapoor [15] also used wave propagation techniques, very similar to Continuous Dijkstra, but with more efficient data structures, to get an $O(n \log^2 n)$ algorithm, n being the number of mesh vertices. Sethian's Fast Marching Method (FMM from now on) was used by Kimmel and Sethian [18] to define a distance function from a source point to the rest of the surface in $O(n \log n)$, and integrate back a differential equation to get the geodesic path. Unlike the others, the last algorithm does not give the exact geodesic paths but an approximation to it; this approximation could be improved using the iterative process we propose in this paper. The algorithm of Chen and Han [5] builds a data structure based on surface unfoldings. In contrast to the other algorithms, it does not follow the wave front propagation paradigm and runs in $O(n^2)$ time with $O(n)$ space. There are many other algorithms to compute shortest geodesics; for more information, we refer the reader to [5, 15, 18, 24, 32] and the references therein.

Chapter overview: In section 2.1 we do a quick review of the definitions for smooth and discrete surfaces. Section 2.2 presents the geodesic algorithm, which is the main contribution of this chapter. We begin with an approximate path and use an iterative process to approach the true geodesic path. Finally in section 2.3 we show some experimental results and adapt our algorithm to the single source shortest path problem.

2.1 Geodesic Curves

Geodesic curves generalize the concept of straight lines for smooth surfaces. Therefore, they have several "good" properties, discussed on section 2.1.1. Unfortunately it is not possible to find such a class of curves over meshes sharing all these properties; as a consequence there are some different definitions for geodesic curves on discrete surfaces, discussed in section 2.1.2, that depend on their proposed use. The rest of this section was mainly extracted from references [10, 26].

2.1.1 Geodesic curves on smooth surfaces

Consider a smooth two-dimensional surface $\mathcal{G}$ and a differential tangent vector field $\mathbf{w} : U \subset \mathcal{G} \to T_P\mathcal{G}$.

Definition 2.1 *Let $\mathbf{y} \in T_P\mathcal{G}$, and consider a parameterized curve $\alpha : (-\varepsilon, \varepsilon) \to U$, with $\alpha(0) = P$, $\alpha'(0) = \mathbf{y}$ and let $\mathbf{w}(t),\ t \in (-\varepsilon, \varepsilon)$ be the restriction of the vector field $\mathbf{w}$ to the curve α. The vector*

obtained by the projection of $(d\mathbf{w}/dt)(0)$ *onto the plane* $T_P\mathcal{G}$ *is called the covariant derivative at* P *of the vector field* $\mathbf{w}$ *relative to the vector* $\mathbf{y}$*. This covariant derivative is denoted by* $(D\mathbf{w}/dt)(0)$.

The definition of the covariant derivative depends only on the field $\mathbf{w}$ and the vector $\mathbf{y}$ and not on the curve α. This concept can be extended to a vector field which is defined only at the points of a parameterized curve. We denote the covariant derivative of a vector field $\mathbf{w}(t)$, defined along a curve α, by $(D\mathbf{w}/dt)(t)$. For details on this subject see [10].

Consider a curve $\gamma : I \to \mathcal{G}$ parameterized by arc length, i.e, $|\gamma'(t)| = 1$ for all t in I. An example of a differential vector field along γ is given by the field $\mathbf{w}(t) = \gamma'(t)$ of the tangent vectors of γ.

Definition 2.2 *γ is said to be geodesic at $t \in I$ if the covariant derivative of γ' at t is zero, i.e.,*

$$\frac{D\gamma'(t)}{dt} = 0;$$

γ is a geodesic if it is a geodesic for all $t \in I$.

The following prop characterizes geodesic curves.

Proposition 2.1 *The following properties are equivalent:*

1. *γ is a geodesic.*
2. *γ is a locally shortest curve; i.e, it is a critical point of the length functional $L(\gamma) = length(\gamma)$.*
3. *γ'' is parallel to the surface normal.*
4. *γ has vanishing geodesic curvature $\kappa_g = 0$* [1].

From item 4 of Proposition 2.1 above, it can be concluded that geodesic curves are as straight as they can be, if we see them from an intrinsic point of view. As a matter of fact, the curve variation up to a second order takes place only in the direction of the surface normal if it has vanishing geodesic curvature. On the other hand, item 2 tells us that a shortest smooth curve joining two points a geodesic. The converse is not true in general: there are geodesic curves which are critical points of the length functional but are not shortest. Nevertheless, the property of being shortest is desirable for curves in many applications and it is perhaps the characterization of geodesic curve more used in practice. Another interesting property of geodesics is that they may have self-intersections, which is impossible for shortest curves.

2.1.2 Discrete geodesics

A curve defined over a mesh will be regular only if it is completely contained in one face or on a set of connected coplanar faces. The existence of such set of connected and coplanar faces is very rare. Therefore, the existence of regular curves passing through more than one face is unlikely. This is the first obstacle that we encounter when trying to generalize geodesics to discrete surfaces. The second one is the fact that it is not possible in general to find a large enough set of curves over discrete surfaces for which all items of Proposition 2.1 hold.

There are some different generalizations of geodesic curves to a discrete surface S, all of them called of *discrete geodesics*. Quasi-geodesics were defined by Aleksandrov and Zalgaller [2] as limit curves of geodesics on a family of converging smooth surfaces. They also defined discrete geodesics as critical points of the length functional over polyhedral surfaces; in other words, they define them as locally shortest curves over S. From now on we call them *shortest discrete geodesics* or simply *shortest geodesics*. In particular, the problem we address in this chapter is to find a shortest geodesic joining two points over a triangular mesh.

Polthier and Schmies [26] defined *straightest geodesics* inspired in the characterization of smooth ge-o-de-sics given by item 4 of proposition 2.1. They defined *discrete geodesic curvature* as a generalization

[1] The geodesic curvature κ_g generalizes to surfaces the concept of curvature of a plane curve. See reference [10].

of the well-known concept of geodesic curvature and straightest geodesics as polygonal curves over S with zero geodesic curvature everywhere. If we call θ the sum of incident angles at a point P of a curve Γ over S and θ_r and θ_l the respective sum of right and left angles (see figure 2.2), the discrete geodesic curvature of Γ at P is defined as

$$\kappa_g(P) = \frac{2\pi}{\theta}\Big(\frac{\theta}{2} - \theta_r\Big).$$

Choosing θ_l instead of θ_r, or changing the orientation of Γ, changes the sign of κ_g. A *straightest geodesic* is a curve with zero discrete geodesic curvature at each point. In particular, straightest geodesics always have $\theta_r = \theta_l$ at every point.

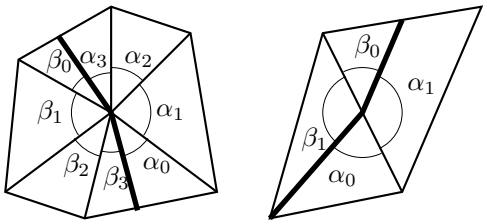


Figure 2.2: Right and left angles (θ_r and θ_l resp.) in a curve. $\theta_r = \sum \alpha_i$ and $\theta_l = \sum \beta_i$.

Let us formalize the concept of discrete geodesic that we are going to use, following [2].

Definition 2.3 *Given two points X and Y on a surface $\mathcal{S}$, a* shortest arc XY *is the shortest of the rectifiable curves joining the points X and Y.*

Definition 2.4 *A curve in which each point has a neighborhood for which the segment of the curve in it is a shortest arc is called a* shortest geodesic*.*

Remark 2.1 The term shortest geodesic may be confusing. Although every shortest arc is always a shortest geodesic, the converse is not true. Note also that a shortest geodesic is a critical point of the length functional, and that a shortest arc is a global minimum, not necessarily unique. Thus, we refer to shortest arcs as optimal shortest geodesics.

The following two lemmas were extracted from [24].

Lemma 2.2 *Let $\mathcal{S}$ be a connected discrete surface. There exists a shortest geodesic from P to any other point $Q \in \mathcal{S}$. Furthermore, among the shortest geodesics between P and Q there exists at least one that is optimal.*

Lemma 2.3 *The general form of a shortest geodesic is a path which goes through an alternating sequence of vertices and edge-crossing subpaths such that the unfolded image of the edge-crossing subpath is a straight line segment, and the left and right angles of the path are greater than or equal to π when passing through a mesh vertex.*

The following proposition explores the difference between straightest and shortest geodesic. It was proved by Polthier and Schmies [26] and will be very useful in defining an strategy to compute a shortest geodesic.

Proposition 2.4 *The concepts of straightest and shortest geodesics differ in the following way:*

1. *A geodesic γ containing no surface vertex is both shortest and straightest.*
2. *A straightest geodesic through a spherical vertex is not locally shortest.*
3. *There exist a family of shortest geodesics through a hyperbolic vertex. Exactly one of them is a straightest geodesic.*

2.2 Geodesic Computation

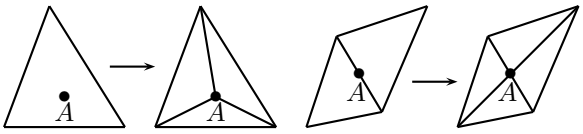
The algorithm that we propose to compute a shortest geodesic Γ between A and B on a surface $\mathcal{S}$ consist of two main steps. First, compute a curve Γ_0 joining A and B; second, evolve it to Γ. These steps are sketched in algorithm 2.1 and the next two subsections explain in detail how to perform them.

Algorithm 2.1 *Compute Geodesic*

Input: A triangular Mesh S, and two points A and B on it.
Output: A discrete geodesic Γ joining A and B.
step 1. Get initial approximation Γ_0
step 2. Iteratively correct $\Gamma_i \ \ (i = 0, 1, \cdots)$ to reach a good approximation Γ_n of Γ

2.2.1 Getting an initial approximation

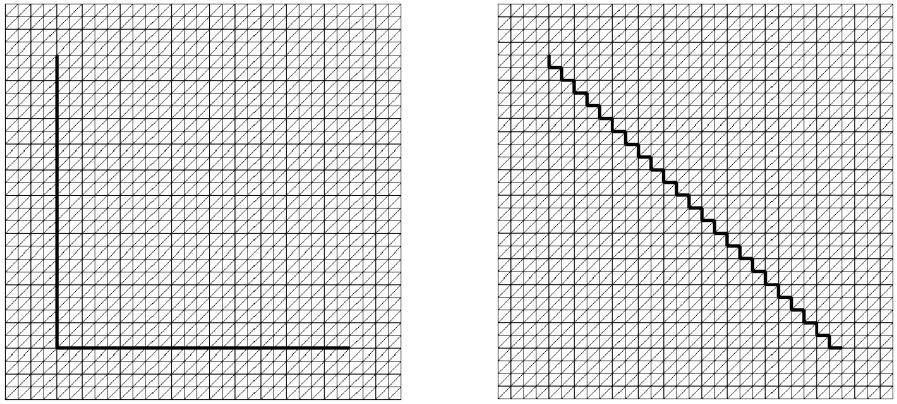
Finding an initial curve Γ_0 over S is rather simple if we consider A and B as vertices, which is not a restriction at all, since we can add them to the set of vertices in an easy manner, see figure 2.3. Thus, in the following, A and B will be treated as vertices.



(a) (b)

Figure 2.3: Inserting the point A in S as a mesh vertex: it belongs to the interior of a face (a) and to an edge (b).

We need an initial polygonal curve Γ_0 joining vertices A and B. The simplest idea is to take some path restricted to the edges. The closer Γ_0 is to the real geodesic Γ, the fewer the number of iterations needed in the second step. Our first attempt was using Dijkstra's Algorithm, but there are some examples where Dijkstra's algorithm may produce a minimum path that is very far from a geodesic one. In figure 2.4 we compare the results of using Dijkstra's Algorithm and FMM to compute Γ_0 in a regular plane triangulation.



(a) (b)

Figure 2.4: First Approximation Γ_0: Dijkstra's Algorithm (a) and FMM (b).

We decided to use FMM to define a distance function in the vertices of the mesh, as done by Kimmel and Sethian [18]. They solve the Eikonal equation

$$|\nabla D| = 1,$$

where $D(P)$ is the (geodesic) distance from A to any point P on S (see references [18, 30] for details). The efficiency of this process relies on the propagation of D over S maintaining a narrow band of vertices close to the front. Once D is computed for every vertex, they must solve the ordinary differential equation

$$\frac{d\chi(s)}{ds} = -\nabla D$$

to get the geodesic path $\chi(s)$. To integrate this equation, using Huen's method, D is approximated in the interior of a face by interpolating a second degree polynomial to the previously computed values of D at the vertices of the face and its three neighboring faces. This process involves some numerical problems and some care must be taken. For instance, the minimum of the interpolant polynomial could be reached in the interior of the face, or the polynomial could be a degenerated quadric. In our implementation, we avoid integration and proceed as follows: place point B in path Γ_0, add to Γ_0 the neighbor of B with minimal distance from A, go on in this way and stop when A is reached. We sketch this process in algorithm 2.2. The correctness of this step is guaranteed since the distance D was defined increasingly from A. Moreover, the same argument permits us to stop FMM once $D(B)$ is computed. The remaining points (where D was not defined) will have $D(P) = \infty$.

Algorithm 2.2 *First Approximation*

Input: A triangular mesh S, and two points A and B on it.
Output: A restricted to edges path Γ_0 joining A and B.
step 1. Compute $D(P)$ for each vertex P in S using FMM
step 2. Put B in Γ_0
step 3. $P_0 = B, \quad i = 0$
while P_i is not equal to A
$P_{i+1} =$ Neighbor of P_i with smaller distance $D(P_{i+1})$ from A.
Put P_{i+1} in Γ_0
$i = i + 1$

Even when we use the whole Kimmel and Sethian's algorithm to compute a shortest geodesic $\widehat{\Gamma}$, it must be corrected since distance computation and integration are performed approximately, and consequently are error-prone. In the next section we describe our strategy to improve the initial approximation.

2.2.2 Correcting a path

Once we have an approximation Γ_i to the geodesic Γ, we need to correct it in order to get a new curve Γ_{i+1} closer to Γ. Since Γ_i is a polygonal line joining A and B, we just have to correct the position of interior nodes, trying to reduce, as much as possible, the length of the curve Γ_i. As Γ has to coincide with a line segment inside every face of S, we restrict the vertices of our successive approximations Γ_0, $\Gamma_1, \Gamma_2 \ \ldots$ to lie on edges or vertices of S.

The path correction step is described in algorithm 2.3. It is inspired in Polthier's straightest geodesics theory, more precisely in the characterization given in proposition 2.4 about the differences between shortest and straightest geodesics.

Algorithm 2.3 *Path Correction*

Input: A triangular mesh S, and a polygonal curve Γ_i joining A and B.
Output: A shorter path Γ_{i+1} joining A and B.
$P_{i+1,0} = P_{i0} = B$ and $P_{i+1,n'} = P_{in} = A$
for $j = 1, 2, \cdots, n-1$
correct the position of node P_{ij}

Each node P_{ij} is corrected to the new position $P_{i+1,j'}$ such that the curve segment between nodes $P_{i+1,j'-1}$ and $P_{i,j+1}$ goes to the geodesic joining them, restricted to the set of faces that are neighbors to P_{ij}.

Note that we use $P_{i+1,j'-1}$, the last corrected node, instead of $P_{i,j-1}$. Doing that, we do not need extra space to store the new curve, since it is enough to substitute P_{ij}'s position by its new position $P_{i+1,j'}$. We will also get a better correction $P_{i+1,j'}$ since, to compute its position, we use a node whose position was previously corrected. Finally, we are able to prove that our algorithm actually reduces the length of Γ_i at each step, what cannot be guaranteed using the old neighbor $P_{i,j-1}$.

In some cases (when P_{ij} coincides with a mesh vertex) it is necessary to replace node P_{ij} by more than a node when correcting Γ_i. Also, some nodes may be eliminated when approaching a mesh vertex. These facts justify the notation $P_{i+1,j'}$ used above for the corrected node in curve Γ_{i+1}.

We will use different procedures to correct the positions of the nodes of the polygonal Γ_i which belong to the interior of mesh edges and of those coinciding with mesh vertices, since they do not behave in the same way. For instance, a point belonging to an edge has only two adjacent triangles while a vertex may have any number of them.

Correction of a node in the interior of an edge.

Suppose the polygonal curve Γ_i is given by the sequence of nodes $B = P_{i0}, P_{i1}, \ldots, P_{in} = A$. For a node P_{ij} ($j \in \{1, 2, \ldots, n-1\}$) lying in the interior of an edge $\mathbf{e}$, we wish to correct its position in order to get a shorter curve Γ_{i+1}. To do that, we unfold the two triangles adjacent to $\mathbf{e}$, and define the new $P_{i+1,j'}$ as the intersection point of $\mathbf{e}$ with the line joining $P_{i+1,j'-1}$ and $P_{i,j+1}$ (see figure 2.5 (a)).

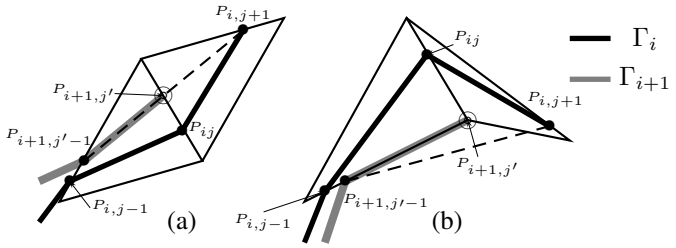


Figure 2.5: Correcting the node position on an edge. Intersection inside the edge (a), and outside (b). Corrected nodes are marked with the symbol ⊙.

Sometimes there is not an intersection point, as in figure 2.5 (b). In such cases we replace P_{ij} by the vertex of $\mathbf{e}$ which is closer to the intersection point between the line containing $\mathbf{e}$ and the line passing by $P_{i+1,j\prime-1}$ and $P_{i,j+1}$.

In both cases the corrected node $P_{i+1,j'}$ gives the shortest curve passing by $P_{i+1,j'-1}$ and $P_{i,j+1}$ inside the two triangles sharing edge $\mathbf{e}$.

Correction of a node coinciding with a mesh vertex.

When P_{ij} coincides with a mesh vertex, the correction is not so simple as in the previous case. Notice that, now, P_{ij} usually belongs to more than two triangles. We need to find a shortest path between $P_{i+1,j'-1}$ and $P_{i,j+1}$ in the union of all triangles containing P_{ij} as vertex. Suppose P_{ij} corresponds to the k^{th} vertex of $\mathcal{S}$; then, such union of triangular faces, the star of vertex k, will be denoted by $\mathcal{S}_k$. For simplicity $P_{i+1,j'-1}$ and $P_{i,j+1}$ are supposed to be on the boundary of S_k; otherwise one of them belongs to the interior of $\mathcal{S}_k$ and in that case we can eliminate it from Γ_i without any loss of information. In fact, this node elimination will result in a shorter curve (see figure 2.6).

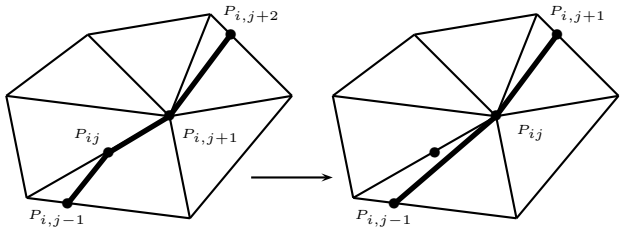
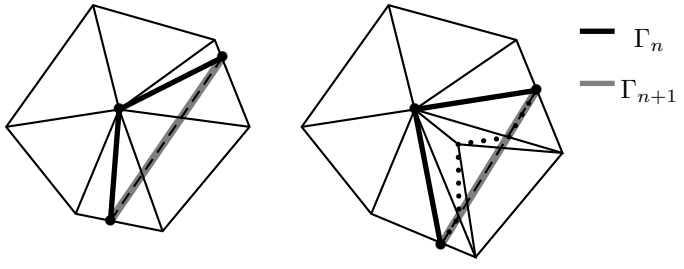


Figure 2.6: Elimination of a node inside $\mathcal{S}_k$.

We first classify node P_{ij} as in definition 1.2 by computing left and right angles θ_l and θ_r, and then we shorten the curve by taking into account proposition 2.4. If P_{ij} is euclidean then $\mathcal{S}_k$ can be isometrically unfolded to be part of a plane and we just have to join the images of $P_{i+1,j'-1}$ and $P_{i,j+1}$ in the unfolding of $\mathcal{S}_k$. If P_{ij} is spherical then no shortest curve may pass through it; in this case choose the part of $\mathcal{S}_k$ with smaller angle, flatten it up, and join $P_{i+1,j'-1}$ to $P_{i,j+1}$. Finally, when P_{ij} is hyperbolic we have two cases. The first one occurs when θ_r and θ_l are both larger than π. In this case no correction is needed, since the curve cannot be shortened by moving P_{ij} (see proof of proposition 2.4 [26]). If one of the angles, say θ_r, is smaller than π then the geodesic must pass through the corresponding side of $\mathcal{S}_k$; we then proceed to flatten it up an compute the line joining $P_{i+1,j'-1}$ and $P_{i,j+1}$.

In all cases we have to compute the intersections of the computed line with the edges of the corresponding flattened part of $\mathcal{S}_k$ and we have to insert them in the polygonal curve in the correct order. Like in previous case, it could happen that the intersection point is outside of some edge (see figure 2.7); in that case we insert the vertex of the edge in the path as we did before. Doing that, we usually obtain a path which is not the shortest one inside the star of P_{ij}. It can be improved performing a new node correction on the extreme of the corresponding edge, obtaining the shortest path between the neighbors of P_{ij}[2].



(a) (b)

Figure 2.7: Correcting a node coinciding with a mesh vertex. Convex star (a) and non-convex star (b). The resulting path before the additional correction in the non-convex star is shown as a dotted line.

Remark 2.2 Note that, in all cases, it is sufficient to trace the geodesic line in the side of $\mathcal{S}_k$ with smaller angle. At hyperbolic vertices, the geodesic can only be traced on the smaller angle's side since the other side has angle bigger than π and the curve cannot be shortened in it. On the other hand, it is possible

[2]Note that in this case the correction has "gone" outside the star of P_{ij}.

sometimes to shorten the curve in both sides of an spherical vertex, but the law of the cosines ensures that the shortest one is obtained in the side with smaller angle. Although the problem of selecting the right side of $\mathcal{S}_k$ to look for $P_{i+1,j'}$ seems not to be necessary at euclidean vertices, the shortest path should also pass through the side with the smaller angle. In order to be convinced of this fact, suppose that $\mathcal{S}_k$ is part of a plane (otherwise we can flatten it isometrically), and consider the triangle formed by $P_{i+1,j'-1}$, P_{ij} and $P_{i,j+1}$; the angle in P_{ij}, which is an interior angle of a triangle, must be less than π, hence the smallest between θ_r and θ_l, since their sum is 2π.

2.2.3 Implementation issues

Is it necessary to add new vertices to $\mathcal{S}$?

To simplify the exposition, we supposed that both extremes of a geodesic are vertices of the mesh $\mathcal{S}$. We suggested to add, if necessary, new vertices to $\mathcal{S}$. However, this is not necessary at all. In fact, we can perform all computations without adding any new vertex to $\mathcal{S}$; lets see:

Defining the initial curve Γ_0: The definition of the distance function D begins by setting $D(A) = 0$, and them propagating a wave until $D(B)$ is set. If A lies in the interior of a face f, we begin setting D at the vertices of f as the euclidean distance $D(V) = \|V - A\|_2$ from A to V, and then propagate the wave. If A lies in the interior of an edge $\mathbf{e}$, we begin setting D at all the vertices of the two neighboring faces of $\mathbf{e}$. The propagation stops when the distance function D have been set at one of the vertices which are neighbors of B. As for point A, there are three neighbors if B lies inside a face, or four if B lies inside an edge. Finally, the first approximation is computed as in algorithm 2.2, but beginning at the neighbor of B with smaller value of the distance function D, and stopping when a neighbor of A was reached. After that we add B and A to Γ_0 as first and last point respectively.

Correcting points inside a face: Although the nodes of geodesics curves are restricted to the edges and vertices of the mesh, some times may be interesting to consider curves having nodes in the interior of faces. This happens, for instance, in the computation of geodesic Bézier curves in chapter 3 where nodes in the interior of faces are obtained due to a subdivision process. To avoid adding new vertices to the mesh we must provide a rule to correct nodes in the interior of faces. Fortunately, this rule is very simple: as a geodesic must be a straight line in the interior of faces, whenever we need to correct a node inside a face we just eliminate it from the curve.

Stopping criterion

An iterative process should always be controlled by a stop criterion, usually based on some error measure. Perhaps the most natural error measure for geodesic computation is given by curve length. However, the difference between the lengths of two successive approximations could be very small even when the curve is far from a shortest geodesic. This behavior is due to the fact that the evolution of the curve has small variation close to mesh vertices, what is usually solved in a second iteration step. In our implementation, we define a measure of error for each curve node based on proposition 2.4, and then define a curve error measure as the maximum node error. For nodes lying in the interior of mesh edges and nodes coinciding with euclidean mesh vertices, we define the error as the discrete geodesic curvature κ_g, since in those cases shortest and straightest geodesics coincide. For nodes coinciding with spherical mesh vertices we define the error as a huge value, since no shortest geodesic can pass through it. For nodes coinciding with hyperbolic mesh vertices we define the error as zero if both θ_l and θ_r are greater than π and as a huge value otherwise, because only in the first case a shortest geodesic can pass through it.

Boundary handling

In some cases a geodesic path can touch a boundary or even coincide in part with it. For example in a plane with a hole or a non convex polygon as boundary, see figure 2.8. In order to correctly handle

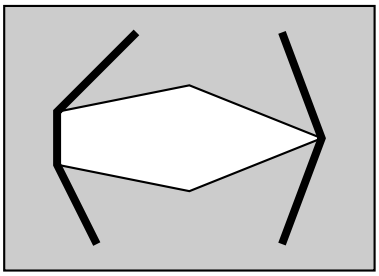


Figure 2.8: Geodesics touching the boundary.

those cases we must take some care close to the boundary. A very simple and effective way to overcome this problem is to try every boundary vertex as an hyperbolic one. The curve Γ_i divides the star of the boundary vertex P_{in} into two regions, one of them containing the boundary, we assign 2π to the angle (θ_r or θ_l) corresponding with that side of the curve. With this simple procedure, we update a boundary vertex only if the side of Γ_i interior to the surface has angle smaller than π, otherwise it remains the same since Γ_i cannot be improved in a neighborhood of P_{in}. In figure 2.9 are shown some geodesics touching the boundary in the bunny model.

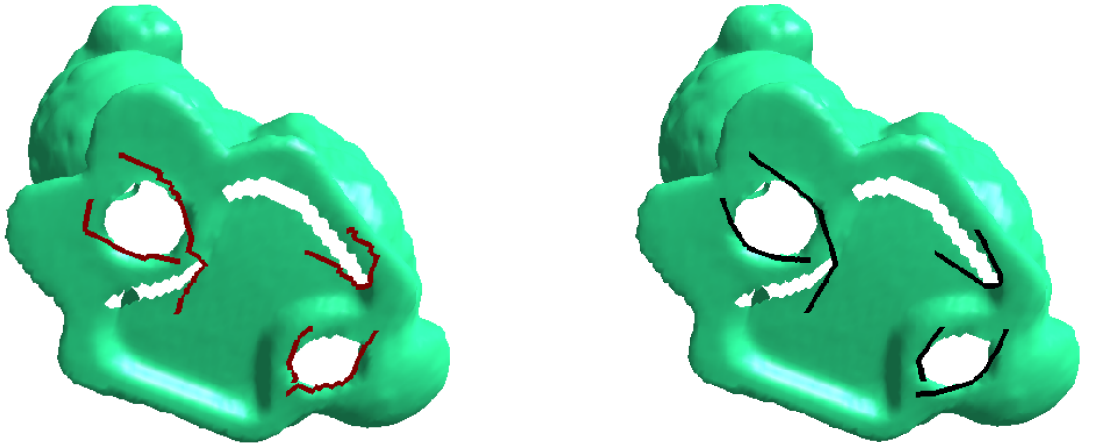


Figure 2.9: Geodesics touching the boundary in the bunny model. Right: Initial approximations. Left: The geodesics.

Speeding up convergence.

In order to improve the performance of our algorithm, we explore a different correction strategy. We put all the interior nodes of the curve Γ_0 in a heap sorted by the error. In each step the node with the largest error is corrected, then it and its neighbors are updated in the heap.

This strategy is particularly useful if part of the path Γ_0 is close to be a geodesic between two intermediate vertices, but it is far from the real geodesic between the extremes A and B of Γ_0. This is the case of geodesic Bézier curves, see chapter 3.

Table 2.1: Run time and number of node corrections for the originally proposed method and the alternative method using a heap.

	Num. of Nodes	Original Method		Using a Heap	
		changes	time	changes	time
1	49	60649	0.60	38089	0.65
2	29	37021	0.37	20469	0.35
3	46	126303	1.22	48478	0.82
4	103	1291478	12.58	549298	9.37
5	126	481039	4.85	80585	1.40
6	75	161026	1.60	51123	0.88

To test the originally proposed algorithm against the strategy proposed in this section we measured (see table 2.1) the number of node corrections and the run time (in seconds) using each method. The program stopped when the error was smaller than 0.005. The number of node corrections was considerably smaller using a heap; as a consequence run times were smaller most of the times.

Run times were measured in an "Intel Pentium 4 CPU 1.60GHz" running Fedora Core 1.

2.2.4 Convergence

Consider the sequence $L(\Gamma_i)$ of curve lengths. A lower bound on the set $\{L(\Gamma_i),\ i = 0, 1, 2, \cdots\}$ is given by 0 since the length of Γ_i must be always positive. On the other hand, the length of Γ_i is reduced at every vertex correction, so we have the inequalities

$$L(\Gamma_0) \geq L(\Gamma_1) \geq \cdots \geq L(\Gamma_{i-1}) \geq L(\Gamma_i) \geq \cdots \geq 0,$$

hence the sequence $L(\Gamma_i)$ converges. Based on this fact and considering that Γ_{i+1} is not allowed to be far from Γ_i (Γ_{i+1} lies in the union of triangular faces touching Γ_i), we have an indication that our method converges to a curve $\widehat{\Gamma}$ which is very close to a shortest geodesic, i.e., to a local minimizer of the length functional. In chapter 4 we do a deeper study of the convergence of algorithm 2.1. A natural question is whether $\widehat{\Gamma}$ is also a global minimizer. As in many other optimization problems, global optimality depends on the initial approximation Γ_0. The curve Γ_0 given by FMM usually happens to be a good initial approximation (see figure 2.1), and we can expect our final curve to be very close to a global minimizer of the length functional.

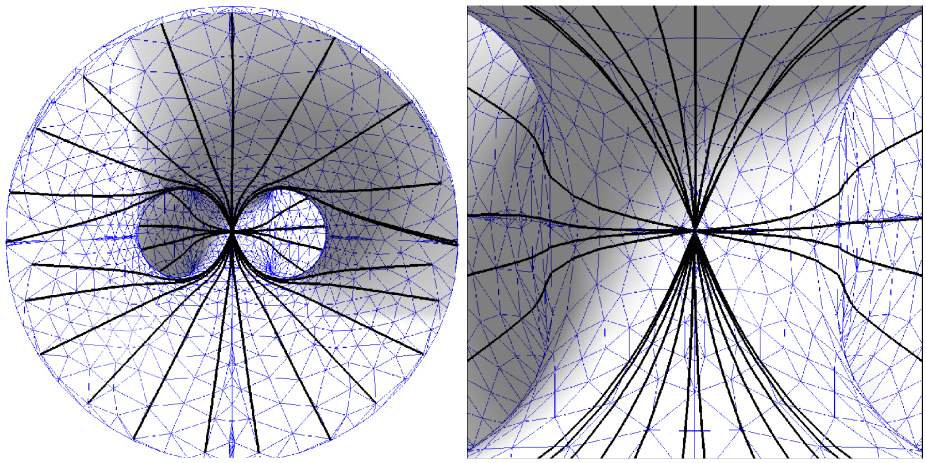
2.3 Experiments

In this section we show some results of our algorithm. In figures 2.1, 2.8, 2.10 and 2.11 we show some geodesics over the Stanford bunny, Costa's surface and a face model. In figures 2.1 and 2.8 we also show the first approximations Γ_0 for the geodesics in the bunny model.

2.3.1 Single source problem

Closely related with our algorithm is the single source shortest path problem. This problem consists of computing a shortest path from a source point A to every point in the surface S.

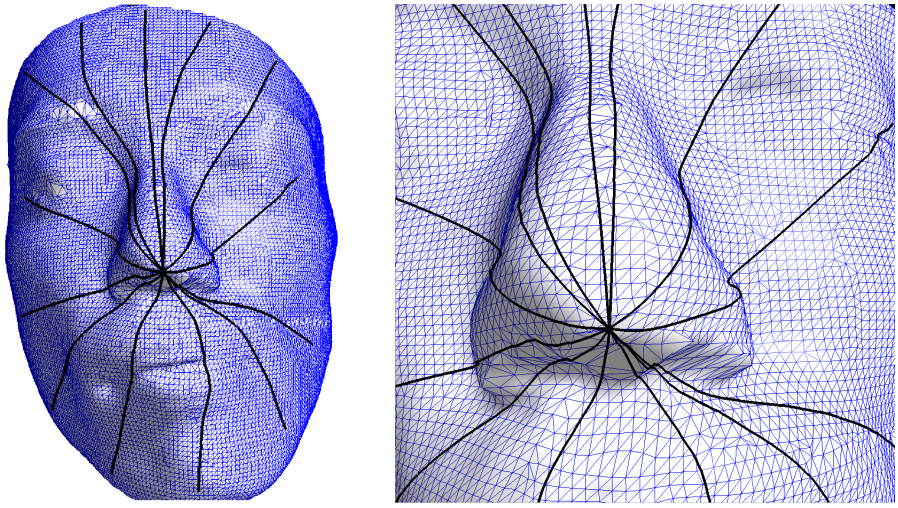
To extend our algorithm in order to solve the single source problem is straightforward. We just need run the distance approximation given by FMM until it has been computed for all surface points instead of stopping when a target point B is reached as done in step 1 of algorithm 2.1. After that, step 2 in algorithm 2.1 must be performed for each point of S.



(a) (b)

Figure 2.10: Some geodesics over Costa's Surface, all sharing a common extreme (a), and a zoom close to the common extreme (b).

Figure 2.12 shows some examples of the application of this algorithm to compute a geodesic from a vertex to all other vertices in a sphere and a simplified Stanford bunny mesh. Notice that no two curves cross over, which is a necessary condition for them to be shortest geodesics. This gives an indication of the correctness of the algorithm.



(a) (b)

Figure 2.11: Some geodesics over a Face model, all sharing a common extreme (a), and a zoom close to the common extreme (b).

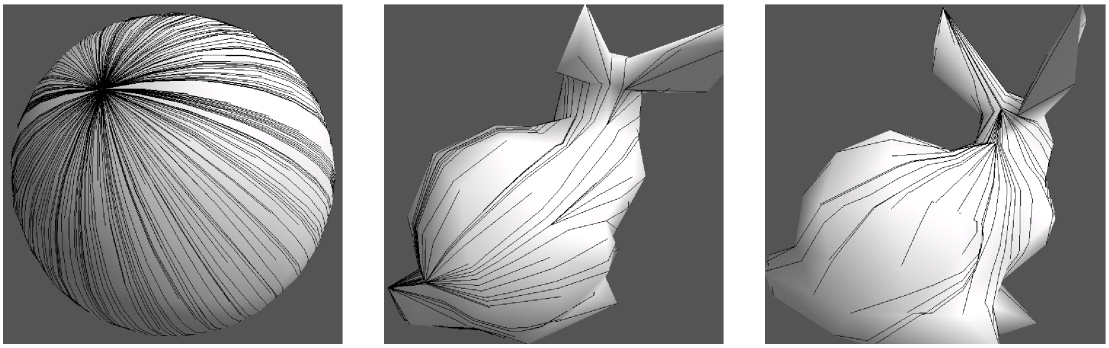


Figure 2.12: Using our algorithm to solve single source problem.

3 Geodesic Bézier Curves

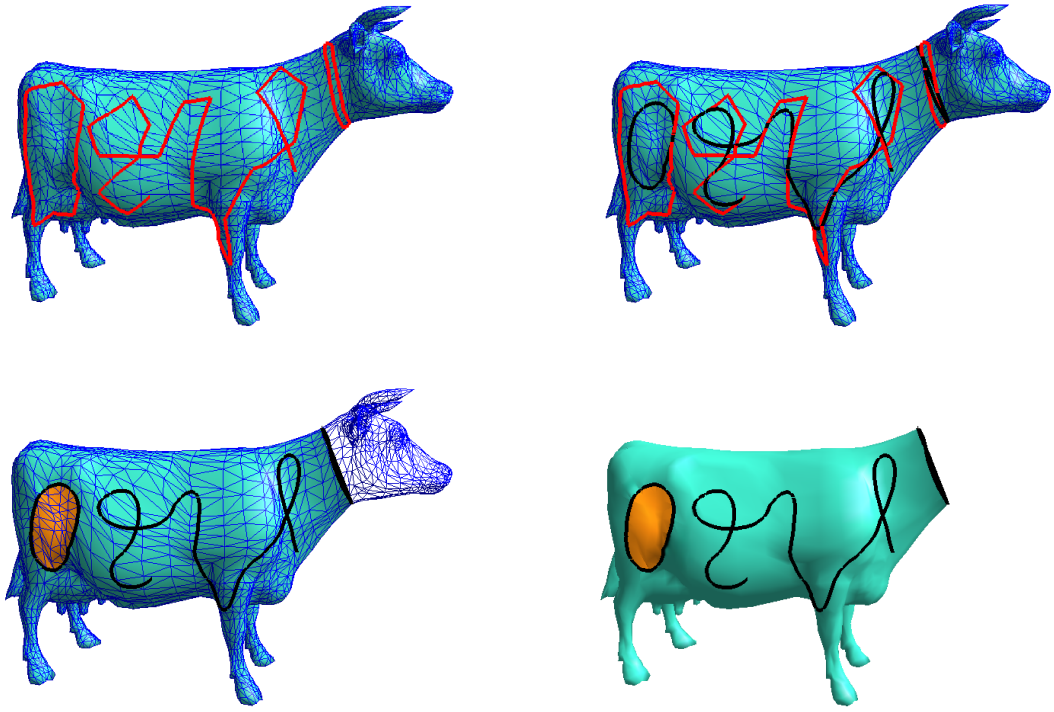


Figure 3.1: Modeling on the surface of a cow. From left: the control polygons of two closed curves and a C^1 spline, the corresponding curves, a region is filled with a different color and other is trimmed, final result.

Designing free-form curves is a basic operation in Geometric Modeling. Doing so in Euclidean space is a widely studied problem; see [7, 11] and many others. The problem becomes harder, however, if we wish to design on a curved geometry, such as triangulated surfaces. Most existing work for the later task, relies on imposing a suitable parameterization, which is usually an unintuitive approach that leads to a series of "trial and error" operations. We pursue instead a direct design in the geometry of the surface, an approach that has received much less attention.

In this chapter we introduce a new class of curves that are suitable for free-form modeling directly on the geometry of a manifold mesh. Defined by means of the de Casteljau Algorithm, they are a natural generalization of Bézier curves. Thus, we call them "Intrinsic Bézier Curves".

Chapter overview: This chapter begins, in section 3.1, with a summary of related work. After summarizing classical Bézier curves theory, we define in section 3.2 the new class of curves and compare them to the classical ones. A study of their use in modeling operations is done in section 3.3. Section 3.4 describes the construction of piecewise Bézier spline curves on triangulations.

3.1 Related Work

The de Casteljau algorithm has been adapted to Riemannian manifolds using geodesic interpolation [25]. However, it has been applied only to some surfaces where geodesic computation is relatively easy, such as spheres or Lie groups; see [8, 29] and the references therein for details. The only modeling operation studied in those works is the construction of cubic splines. Furthermore, trimming seems to be very hard to perform in this setting. We define curves using the same idea as in [25], but we consider manifold triangulations.

Another generalization of cubic splines to smooth manifolds is given in [14, 28]. The curves are defined by interpolating a set of knot points on the surface, minimizing an energy functional. These results are also applicable to triangulations, but they require the computation of local smooth approximations. If the position of a knot is changed, the whole curve must be recomputed since there is no local control of its shape. On the other hand, the interpolation of tangent vectors or higher derivatives has not been studied. Using geodesic Bézier curves we can overcome these difficulties, and it becomes possible to prescribe derivatives at knot points and to have local control of the segments of the Bézier spline curve.

Trimming is a very important application in CAGD. This operation is usually done by means of a parameterization. One has to figure out which curve in the parameter space corresponds to the desired curve on the surface, which is generally a difficult problem. Most of the time these curves are obtained as the result of CSG operations, or more general surface intersections [20, 6]. An approach for subdivision surfaces is to modify the original (coarse) mesh in order to obtain a trimmed limit surface [4, 22]. Our trimming operations are done directly on the mesh and no parameterization is needed.

Recent works [35, 34, 33] define a general framework for curve subdivision schemes. Geodesic Bezier curves fits into this framework. However, smoothness of limit curves is only studied – and proved – in the case of smooth manifolds, considering meshes as an approximation of smooth surfaces, what is not always the case. For example, a potential user may be interested in modeling on a coarse mesh instead of a refined one. Besides, models with sharp features are best approximated with non-smooth surfaces. In this thesis we analyze the smoothness of geodesic Bézier curves in the context of triangular meshes, and also how to handle modifications in the position of control points at interactive rates, what is not done in those works. Our results are also applicable to the other geodesic-based subdivision schemes fitting into this framework.

3.2 Geodesic Bézier Curves

Bézier curves are of great importance when modeling on $\mathbb{R}^n$. A natural question is how to generalize them to curved geometries. In this section we propose a class of curves that generalize Bézier curves to manifold triangulations.

3.2.1 Classical Bézier Curves

Given n+1 control points $P_0, P_1, \ldots, P_n$ in $\mathbb{R}^d$, they define a curve given by the following parametric expression:

$$P(u) = \sum_{i=0}^{n} \binom{n}{i} (1-u)^{n-i} u^i P_i, \quad 0 \leq u \leq 1, \tag{3.1}$$

P is known as Bézier curve of degree n, and the set of control points $P_0, P_1, \ldots, P_n$ forms its control polygon. Note that P interpolates the two extreme control points P_0 and P_n, being tangent to the control polygon at these points. It also "imitates" the form of the control polygon, making the task of designing with Bézier curves very intuitive. That's the reason why Bézier curves are so popular for CAD/CAGD applications. More information about this subject can be found in [7, 11]; figure 3.2 shows an example of a Bézier curve of degree 3.

The de Casteljau Algorithm [9] provides a geometric procedure to evaluate a Bézier curve at any parameter $u \in [0, 1]$, using repeated linear interpolation:

Algorithm 3.1 *de Casteljau*

Input: The control points $P_0, P_1, \ldots, P_n$ and a parameter $u \in [0, 1]$
Output: The point $P(u)$.
step 1. for $i = 0, \ldots, n$ **set** $P_i^{[0]}(u) = P_i$.
step 2. for $j = 1, \ldots, n$
for $i = j, \ldots, n$
$P_i^{[j]} =$**interpolate**$(P_{i-1}^{[j-1]}(u), P_i^{[j-1]}(u), u)$.
step 3. $P(u) = P_n^{[n]}$

In step 2 we use the function **interpolate**(A, B, u), which performs a linear interpolation between A and B with parameter u: **interpolate**$(A, B, u) = (1-u)A + uB$.

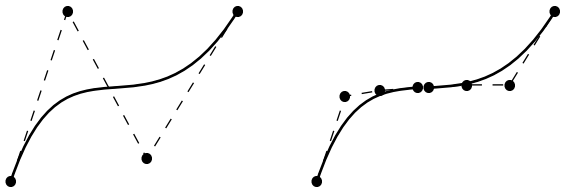


Figure 3.2: A subdivision step of a control polygon and its Bézier curve.

From de Casteljau's algorithm one can define a subdivision scheme whose limit curve is the Bézier curve given by the control polygon. Given a parameter value u and a control polygon, we can obtain two new control polygons for the segments $P([0, u])$ and $P([u, 1])$:

Algorithm 3.2 *Subdivision of a control polygon*

Input: The control points $P_0, P_1, \ldots, P_n$ and a parameter $u \in [0,1]$
Output: Two sets of control points defining $P([0,u])$ and $P([u,1])$.
step 1. deCasteljau$((P_0, P_1, \ldots, P_n), u)$.
step 2.

$$P([0,u]) = \textbf{bezier}(P_0^{[0]}, P_1^{[1]}, \ldots, P_n^{[n]})$$
$$P([u,1]) = \textbf{bezier}(P_n^{[n]}, P_n^{[n-1]}, \ldots, P_n^{[0]})$$

Evaluating the curve at u, using de Casteljau's algorithm, provides the intermediary interpolated points $P_i^{[j]}$. The output (step 2) are the control polygons defining both (Bézier) segments of the curve. Figure 3.2 shows a subdivision step of the control polygon of a degree 3 Bézier curve.

Algorithm 3.2 provides the rule for the subdivision scheme converging to the curve. Additionally, this scheme can be made adaptive by stopping the subdivision whenever a control polygon can be considered as "almost straight".

3.2.2 Bézier Curves on Manifold Triangulations

Intrinsic Bézier curves are defined by means of the subdivision algorithm for classical Bézier curves. Given a control polygon $P_0, P_1, \ldots, P_n$ on a surface $\mathcal{S}$, we want to compute a curve $\mathcal{C}$ on $\mathcal{S}$ interpolating P_0 and P_n, whose shape is controlled by the position of the interior points $P_1, P_2, \ldots, P_{n-1}$.

The curve $\mathcal{C}$ is defined as the limit of the subdivision scheme given in algorithm 3.2 of section 3.2.1. The main difference is that the sides of the control polygon are no longer line segments, but geodesics connecting the control points. This imposes the necessity of modifying the interpolation step on algorithm 3.1. The equivalent to linear interpolation in the geometry of the surface is the interpolation along geodesic lines. Algorithm 3.3 describes the interpolation step in the case of manifold triangulations:

Algorithm 3.3 *Interpolation on Manifold Triangulations*

Input: A manifold triangulation $\mathcal{S}$, two points Q_1 and Q_2 on it and a parameter $u \in [0,1]$
Output: A point Q on $\mathcal{S}$ interpolating Q_1 and Q_2.
step 1. $\Gamma =$**ComputeGeodesic**(Q_1, Q_2).
step 2. $Q =$ the point of Γ such that

$$\mathrm{d}_\Gamma(Q_1, Q) = u\mathrm{d}_\Gamma(Q_1, Q_2)$$

In algorithm 3.3, $\mathrm{d}_\Gamma(A,B)$ computes the distance between A and B along Γ. Note that since Γ is a polygonal line, it is very simple to perform step 2.

To compute an approximation of $\mathcal{C}$, we can use the subdivision adaptive algorithm. It stops at some prescribed level of subdivision or when the control polygon can be considered as a geodesic segment; i.e., when all of its control vertices have error smaller than a prescribed tolerance. Figure 3.3 shows some geodesic Bézier curves along with their control polygons.

The use of de Casteljau's algorithm in the definition of geodesic Bézier curves makes them a generalization of planar Bézier curves. Note that a geodesic on a plane is a straight line. Therefore, when the triangulation is planar both concepts coincide; see for example the curves designed on the (planar) faces of the cube in figure 3.3. As in the case of classical Bézier curves, we have a parameterization of our curves with parameter $u \in [0,1]$. Evaluation at any particular parameter value can be performed easily by subdividing the corresponding control polygon at each level of subdivision, imitating the bisection algorithm used to compute roots of polynomials. Previous calculations can be used to evaluate at new parameter values, this can be useful when performing many evaluations.

It is known that shortest geodesics are not unique on triangulations. Consequently, the use of a different subdivision parameter may lead to a different curve. So geodesic Bézier curves depends on the control

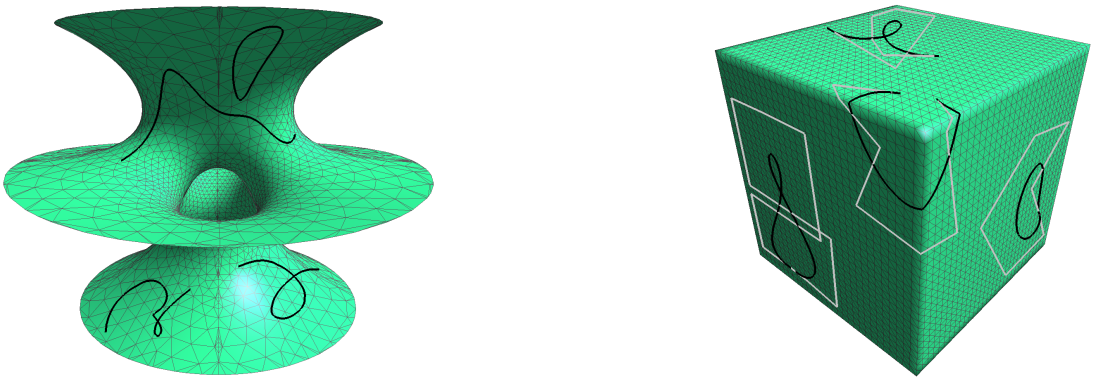


Figure 3.3: Some geodesic Bézier curves. Control polygons are also shown in the Cube model.

points and the chosen subdivision parameter u. To our experience, the curves obtained with different values of u are very close to each other. We are currently studying the theoretical issues related to the choice of of u. In practice, selecting a fixed value of u gives us a subdivision curve. In this work we have chosen $u = 0.5$ and therefore we have a midpoint subdivision scheme. All the figures were generated in this way.

Geodesic algorithm selection

There are several algorithms to compute geodesics (see chapter 2) and any of them could be used both to compute geodesic Bézier curves and to perform user interaction. We chose the algorithm described in chapter 2 for two reasons. In first place, it relies on the correction of an initial curve assumed to be close to the true geodesic. Since de Casteljau's algorithm is a sort of corner-cutting process, a part of each control polygon can be used as initial curve to compute the geodesics needed in the computation of control polygons in the following level of subdivision. On the other hand, during interaction the new control polygons are very close to the previous ones and they can be computed very fast. Using other algorithms as [32] also permits very fast interaction, but at the cost of storing a tree for each control point. The tree associated with a control point must be updated any time its position is changed.

3.2.3 Properties of Geodesic Bézier Curves

Geodesic Bézier curves share some properties with classical ones.

Proposition 3.1 *Geodesic Bézier curves interpolate P_0 and P_n and are tangent to the control polygon at these points.*

Proof: Consider the first line segment of the first side (geodesic line) of the control polygon. It is entirely contained in a face F and for N large enough, the initial sub-polygon at level N is entirely contained in F. So, the corresponding curve segment is a segment of a planar Bézier curve, which is tangent to the first segment of the N^{th} level subpolygon, that is contained on the first side of the 1^{st} level polygon. So, the curve is tangent to its control polygon at P_0; the same applies to P_n. □

The convex hull property of classical Bézier curves is very important. It permits to know where the curve is even before evaluating it. This is specially important when designing using the de Casteljau's algorithm. In this case we can stop subdividing when the control polygon is almost straight. After defining convex hull in chapter 4 we will establish the convex hull property for geodesic Bézier curves. A consequence of the convex-hull property is that geodesic Bézier curves have *linear precision*, i.e, if all the control points are evenly distributed on a shortest geodesic, the resulting geodesic Bézier curve is the same shortest geodesic with parameterization proportional to curve length. If the points are not evenly distributed the resulting curve is also the geodesic but its parameterization is not proportional to the curve length.

Classical Bézier curves has C^∞-continuity. Lets study the smoothness of geodesic Bézier curves:

Proposition 3.2 *A geodesic Bézier curve has (at least) C^1 continuity when intersecting an edge of the mesh.*

Proof: Let s_0 be the parameter value corresponding to the point $P(s_0)$ of $\mathcal{C}$ intercepting the edge e of mesh $\mathcal{S}$. We have two cases: (i) for N large enough, there is a segment $\mathcal{C}_{N,i}$ of $\mathcal{C}$ completely contained in the union of the two faces adjacent to e, and such that $P(s_0) \in \mathcal{C}_{N,i}$; or (ii) case (i) does not hold and $s_0 = k/2^N$, where k is an integer such that $0 < k < N$.

In case (i) $\mathcal{C}$ is C^∞ at $P(s_0)$, since it is a point of a plane Bezier curve defined in the unfolding of e. If case (ii) holds, there is an N large enough such that each of the two polygons corresponding to level N of subdivision is contained in one of the two faces adjacent to edge e. So, in the unfolding of the two faces, the two segments of control polygons at level N with $P(s_0)$ as endpoint are collinear and have the same length. As a consequence the plane spline defined by these two control polygon is (at least) C^1 at its junction point $P(s_0)$, and hence $\mathcal{C}$ is C^1 at $P(s_0)$ too. □

Proposition 3.3 *Each connected component of the intersection of a geodesic Bézier curve with the interior of a mesh face is a C^∞ plane curve, except for (at most) a countable set of points, where it is C^1.*

Proof: Let f be a face of $\mathcal{S}$, and s the parameter value corresponding to the point $P(s)$ of $\mathcal{C} \bigcap f$.

As in the preceding proposition, there are two cases: (i) for N large enough, there is a segment $\mathcal{C}_{N,i}$ of $\mathcal{C}$ completely contained in f and such that $P(s) \in \mathcal{C}_{N,i}$; or (ii) case (i) does not hold and $s = k/2^N$, where k is an integer such that $0 < k < N$.

In case (i) $\mathcal{C}$ is C^∞ at $P(s)$, since it is a point of a plane Bezier curve defined in f. If case (ii) holds, there is an N large enough such that each of the two polygons corresponding to level N of subdivision is contained in f. So, the two segments of control polygons at level N with $P(s)$ as endpoint are collinear and have the same length. As a consequence the plane spline defined by these two control polygon is (at least) C^1 at its junction point $P(s)$, and hence $\mathcal{C}$ is C^1 at $P(s)$ too.

Note that the set of points satisfying condition (ii) is finite in almost all the cases. When $\mathcal{C}$ pass through a vertex of f, this set can be infinite. However, it is countable, since it is a subset of $\{s = k/2^N, \text{ s.t. } k, N \in \mathbb{N}, k < N\}$ which is a countable set. □

As a consequence of previous propositions we have that $\mathcal{C}$ is as smooth as possible in the interior of faces and when crossing a mesh edge. The analysis of the passage of $\mathcal{C}$ through mesh vertices is more complicated and is left as future work.

3.3 Modeling

In order to model with geodesic Bézier curves we must be able to perform the usual modeling operations. Moreover, the user should be allowed to modify a curve at interactive rates. In this section we first describe how to efficiently handle user interaction. Following that, we present a simple algorithm for region fill and trimming.

3.3.1 User Interaction

Fast user interaction is very important in free-form design operations. The user should be able to modify any previously defined curve by changing the position of some of its control points. This operation should be as fast and easy as possible; for example, it must be possible to select and drag any control point using the mouse. Every time the position of a control point is changed, its neighboring sides in the control polygon should be recomputed. These (at most two) sides are geodesic lines and we must recompute them very fast, at least approximately. Each new (recomputed) geodesic is very close to the old (original) one, since one of their extremes remain fixed while the other one is very close to

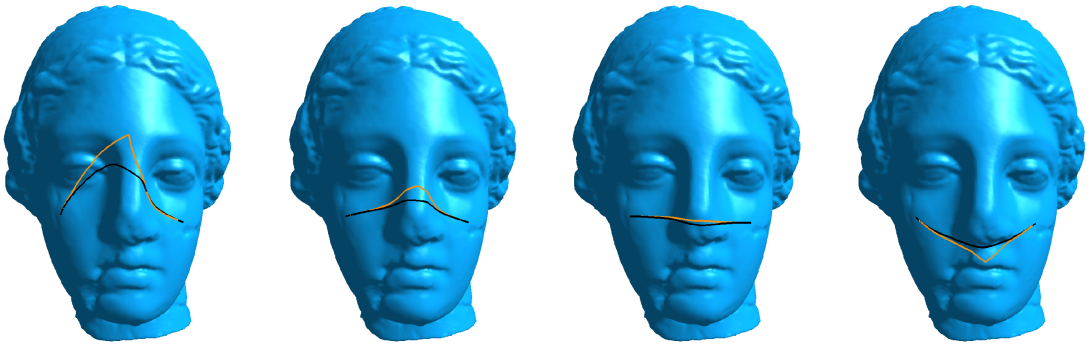


Figure 3.4: Igea model: Four different positions for the middle control point in a curve with three control points.

the corresponding extreme in the old curve. Hence we use, as initial approximation for the algorithm described in section 2.1, the original curve after adding to it the line segment joining its extreme to the new control point position. Because the initial segment is very close to the recomputed one, this update process runs very fast. Additionally, we force the geodesic computation to perform fewer iteration steps during interaction since the user only needs to have a good idea of the shape of the control polygon. When the user releases the mouse, full-precision geodesics are computed and the curve is then recomputed. Figure 3.4 shows three different positions for the middle node during user interaction with a third order curve.

3.3.2 Region Fill and Trimming

We are now concerned with the problem of identifying a piece of a surface $\mathcal{S}$ limited by one or more curves defined on it. Solving this problem allows us to trim (cut) a piece of $\mathcal{S}$, to paint it with a certain color, or to map a texture to it. Given a point P in $\mathcal{S}$, typically obtained by a mouse click, the idea is to use a flood-fill algorithm, propagating a wavefront from this point until it reaches the boundary curves. Algorithm 3.4 describes how to identify the faces in the region $\mathcal{R}$ that contains the point P.

Algorithm 3.4 *Identify region*

Input: A point $P \in \mathcal{S}$
Output: The set $S_{\mathcal{R}} = \{f \in \{\text{faces of } \mathcal{S}\}, \text{s.t.} f \bigcap \mathcal{R} \neq \emptyset\}$
step 1. f = face containing P.
step 2. **push**(f,$L_{\mathcal{R}}$)
step 3. **while** $L_{\mathcal{R}}$ is not empty
g = **pop**($L_{\mathcal{R}}$)
push(g,$S_{\mathcal{R}}$)
for $h \in$ {neighbors of g}
if(**can_propagate**($g \mapsto h$))
push(h,$L_{\mathcal{R}}$)

In algorithm 3.4 above, $L_{\mathcal{R}}$ is an auxiliary list of faces. The function **can_propagate** returns **true** if the following three conditions hold:

1. h does not belong to $L_{\mathcal{R}}$,
2. h does not belong to $S_{\mathcal{R}}$, and
3. $\mathcal{R}$ contains the edge common to g and h, or part of it.

In practice it is not necessary to know if condition 3 holds. Instead we only consider the faces that are adjacent to edges intersecting region $\mathcal{R}$, see figure 3.6. When $L_{\mathcal{R}}$ becomes empty we have in $S_{\mathcal{R}}$ all the faces contained in the interior of $\mathcal{R}$ and also the faces cutting the boundary curves. They are colored red and green respectively in figure 3.5 (left). Once we have identified the set $S_{\mathcal{R}}$ of faces cutting $\mathcal{R}$, we

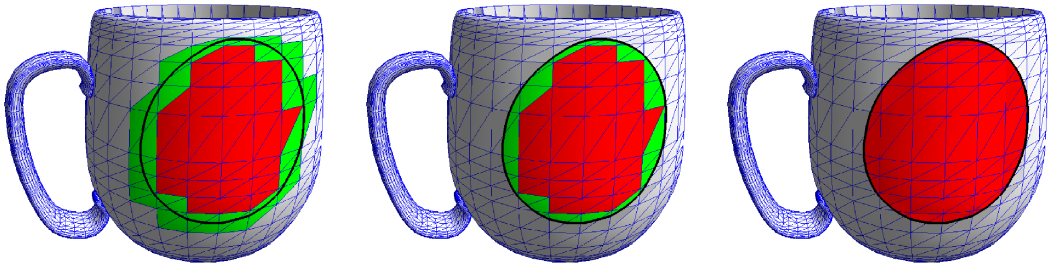


Figure 3.5: Region finding stages. Left: set $S_{\mathcal{R}}$ with boundary faces highlighted. Middle and Right: the region $\mathcal{R}$ after eliminating the part of boundary faces not belonging to it.

Figure 3.6: Propagation directions. Arrows indicate by what edges can the wave be propagated. Bullets indicate what portion of the edges belong to $\mathcal{R}$ (shadowed region).

must decide which part of the boundary faces belongs to $\mathcal{R}$. To do that, during propagation we mark each portion of an edge intersecting $\mathcal{R}$, see figure 3.6. With this information we can decide which part of the planar subdivision defined by the boundary curves in each face belongs to $\mathcal{R}$.

The above described process can easily be performed if the seed point P belongs to a face which is entirely contained in $\mathcal{R}$. If P belongs to a face crossed by the boundary of $\mathcal{R}$, we subdivide it until P is inside an interior face (see figure 3.7). For texture mapping or trimming it is not sufficient to identify

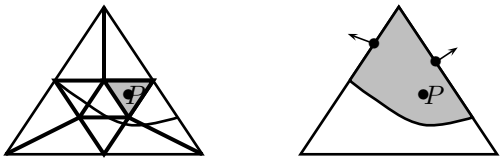


Figure 3.7: Locating seed point

the part of $\mathcal{S}$ (i.e., the region $\mathcal{R}$) selected by the user. It is also necessary to have a model of it. In those cases we can triangulate the corresponding part of each face crossed by the boundary of $\mathcal{R}$. Figure 3.8 shows some regions filled or trimmed in the Cube and the Bunny models.

3.4 Piecewise Bézier Spline Curves

A powerful tool for modeling is the use of piecewise spline curves, allowing local control of the shape of the curve as well as faster computations by means of segments f low degree. We want to compute piece-

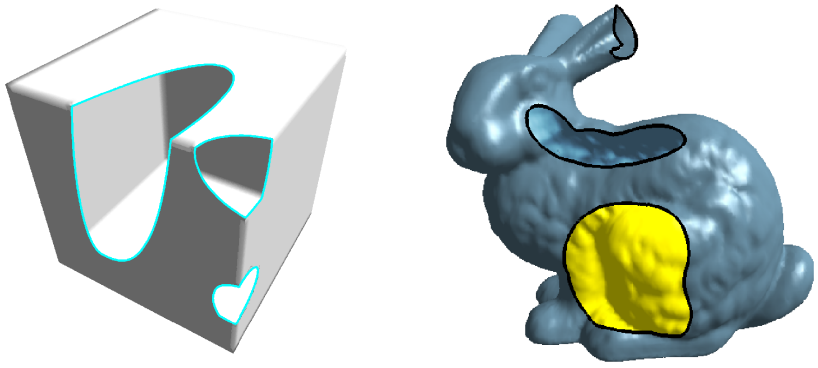


Figure 3.8: Filled and trimmed regions: Trimmed Cube and Stanford's bunny with two trimmed regions and one filled.

wise spline curves of geodesic Bézier curves, so next we investigate how to guarantee some continuity at junction points.

As usual, C^0 continuity is reached by defining the first control point of a segment $\mathcal{C}_{i+1}$ to be the same as the last control point of its previous segment $\mathcal{C}_i$. To guarantee C^1 continuity is harder because we must have the last side of the control polygon of $\mathcal{C}_i$ aligned with the first side of the control polygon of $\mathcal{C}_{i+1}$. Moreover, the length of these two sides must be the same. This means that we need to locate three control points in the same geodesic line. In other words, the position of the two first control vertices of the segment $\mathcal{C}_{i+1}$ are determined by the position of the control vertices of the previous segment $\mathcal{C}_i$.

Given the control polygon of the i^{th} segment $\mathcal{C}_i$ of a spline curve $\mathcal{C}$, how to compute the two first control points P_0^{i+1} and P_1^{i+1} of $\mathcal{C}_{i+1}$? Its first control point P_0^{i+1} is the same as the last control point P_m^i of $\mathcal{C}_i$. The second one, P_1^{i+1}, is hard to find because we do not know how to continue the geodesic line between P_{m-1}^i and P_m^i. For smooth surfaces we can compute the unique geodesic passing by a point in a direction. This is not the case for shortest geodesics on meshes. Nevertheless the straightest geodesics defined by [26] have this nice property. For that reason we define the first side of the control polygon of $\mathcal{C}_{i+1}$ as the straightest geodesic continuing the last side of the control polygon of $\mathcal{C}_i$. It is known [26] that if a straightest geodesic does not pass by a spherical vertex, it is also a shortest geodesic. So we can expect that most of the times our control polygon will be defined by means of shortest geodesics. It is important to note that all the properties of section 3.2.3 are also satisfied if we replace one or more of the shortest geodesics by straightest ones. Thus, the relaxation we did to the definition of the control polygon, in order to have C^1 continuity, is more than justified.

Finally note that modifying the position of P_{m-1}^i modifies the position of P_1^{i+1} and vice versa. In the last case, the last side of the control polygon of C_i will be a straightest geodesic. Modifying the position of the junction point conduce us to modify at least the position of one of the control points P_{m-1}^i and P_1^{i+1}. Figure 3.9 show the use of C^1 splines to write in the surface of the Stanford's bunny model. The middle curve in figure 3.1 is a C^1 spline, composed by 8 Bézier segments.

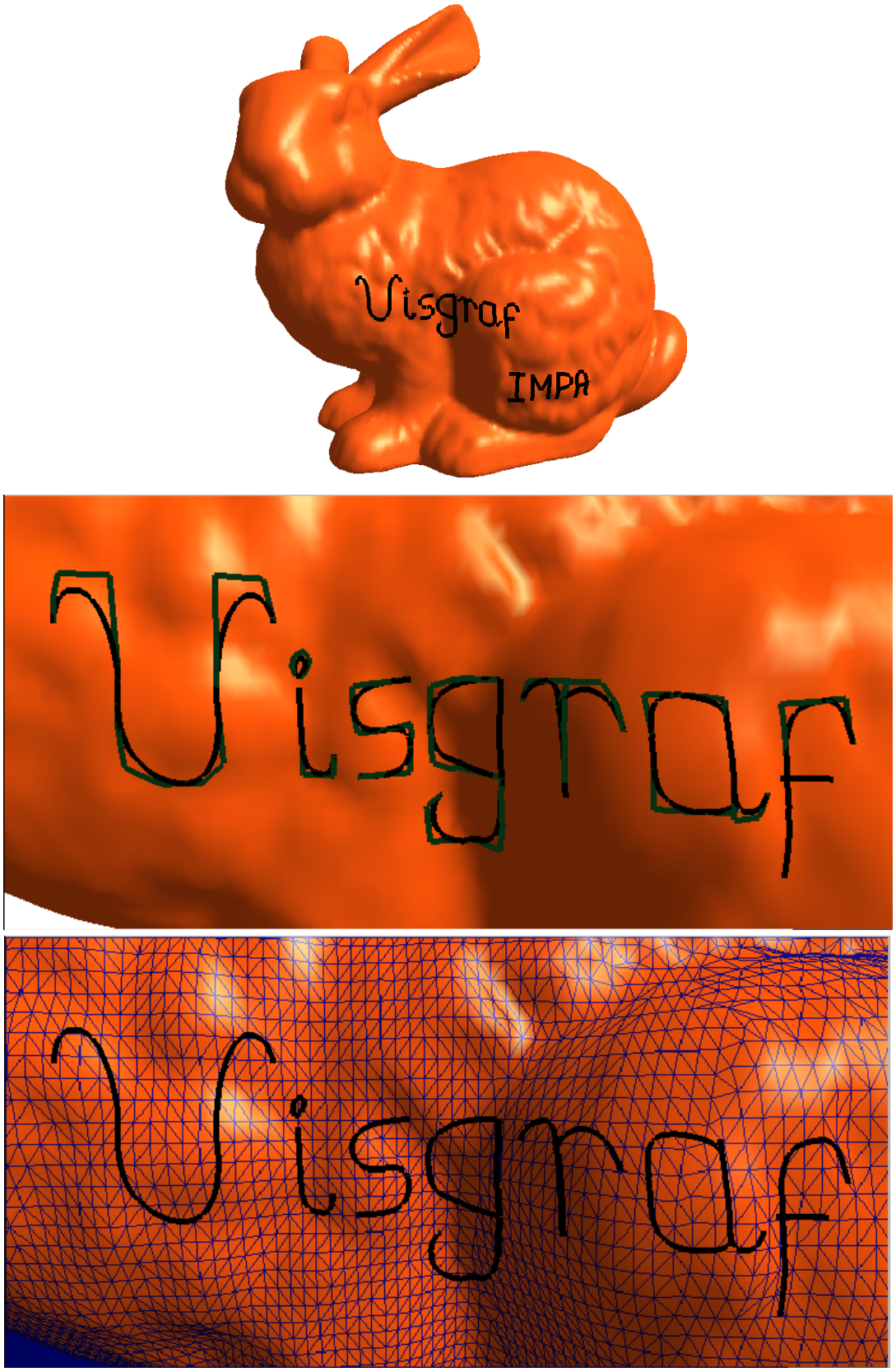


Figure 3.9: Splines on the surface of the bunny.

4 Convex Sets on Discrete Surfaces

To define convex sets in triangulations requires some care. A naive definition, as the intersection of the surface with a convex set, will not work because it ignores the intrinsic geometry of the surface. As in the case of geodesics, there are different definitions obtained by imitating the behavior of plane convex sets [2], but they are too restrictive. For example, non-optimal shortest geodesics are not convex in the usual sense. Our definition allow them to be convex.

In this chapter we define convex sets based on the geodesic curvature of the boundary curves, and derive the concept of convex hull in manifolds. Those definitions are very useful to prove the convergence of the geodesic algorithm of chapter 2, and the convex hull property of geodesic Bézier curves defined in chapter 3.

4.1 Discrete Geodesic Curvature

Polthier and Schmies [26] defined *straightest geodesics* as curves with zero geodesic curvature $\kappa_g(P)$, where

$$\kappa_g(P) = \frac{2\pi}{\theta}\Big(\frac{\theta}{2} - \theta_r\Big).$$

In order to obtain a similar characterization for shortest geodesics, we must employ an alternative definition of discrete geodesic curvature:

Definition 4.1 *The* discrete geodesic curvature *of a polygonal curve at a given point P is given by the following expression.*

$$\kappa_s(P) = \begin{cases} 0, & \textit{if } \theta(P) > 2\pi,\ \ \theta_r(P) \geq \pi \ \textit{ and } \ \theta_l(P) \geq \pi \\ \infty, & \textit{if } \theta(P) < 2\pi \ \textit{ and } \ \theta_r(P) = \theta_l(P) \\ \kappa_g(P), & \textit{otherwise} \end{cases}$$

Remark 4.1 We use the notation κ_s to distinguish this new definition of discrete geodesic curvature from Polthier's one. The subindex s stands for shortest.

Remark 4.2 Boundary points are treated as hyperbolic, as done in section 2.2.3.

With this alternative definition of geodesic curvature at hand, we can characterize discrete shortest geodesics on triangulations:

Proposition 4.1 *A polygonal curve Γ contained in a manifold triangulation $\mathcal{S}$ is a shortest geodesic if and only if its geodesic curvature κ_s is identically zero.*

Proof: Let Γ be a shortest geodesic on $\mathcal{S}$ and P a node of Γ. If P does not coincide with a mesh vertex, or coincides with an euclidean one, then θ_r and θ_l are equals; consequently $\kappa_s(P) = \kappa_g(P) = 0$. If P

coincides with a hyperbolic vertex then both θ_r and θ_l must be greater than π, hence $\kappa_s(P) = 0$. Finally, P cannot coincide with a spherical vertex.

On the other hand, suppose that Γ have zero geodesic curvature everywhere. That means –see definition 4.1 and proposition 2.4– that a small perturbation in the position of any vertex will increase the length of Γ. Thus, it is a local minimum of the length functional and consequently a shortest geodesic. □

4.2 Convex Sets

It is known that if the boundary of a plane convex set is a smooth curve, then its curvature does not change its sign [10]. If the boundary is a polygonal line, then at each vertex the angle corresponding to the interior of the set is smaller that the exterior one. Based on these facts, we define convex sets on triangulations.

Definition 4.2 *Let $\mathcal{C}$ be a connected subset of a surface $\mathcal{S}$. $\mathcal{C}$ is* convex *if its boundary $\partial\mathcal{C}$ can be parametrized by a closed curve $\Gamma : I \longrightarrow \partial\mathcal{C}$ such that:*

1. $\forall t \in I \;\; \kappa_s(\Gamma(t)) \leq 0$, *and*
2. *the interior of set $\mathcal{C}$ is always situated in the left side of Γ.*

Remark 4.3 If we replace'$\leq$' by '$\geq$' and 'left' by 'right' in definition 4.2, we get an equivalent definition of convex set.

Remark 4.4 We do not require the curve Γ parameterizing $\partial\mathcal{C}$ to be simple. Otherwise, a simple geodesic curve would not be convex. Figure 4.1 shows two convex sets in a plane with a hole where the boundary curves are not simple.

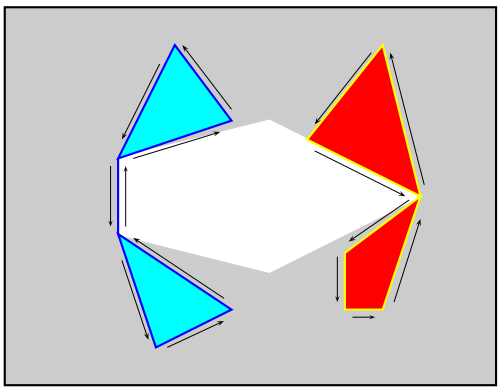


Figure 4.1: In a plane with a hole, two convex sets whose boundary curves are not simple.

Remark 4.5 Convex sets in a plane triangulation (without holes) are also convex in the usual sense.

Unfortunately, the usual properties of planar convex sets do not hold in the general case for convex sets on triangulations. In next section we are going to see an example where the intersection of two convex sets is not connected. Besides, there are convex sets having two different points that cannot be joined by an optimal shortest geodesic. However, a convex set always contains a shortest geodesic, not necessary optimal, joining two any points in it.

Proposition 4.2 *Let $\mathcal{C}$ be a convex set on the surface $\mathcal{S}$ and let A and B be two points of $\mathcal{C}$. Then there exists a shortest geodesic joining A and B that is completely contained in $\mathcal{C}$.*

Proof: Consider the surface $\mathcal{S}'$ defined by the set $\mathcal{C}$. There exists a geodesic Γ joining A and B on $\mathcal{S}'$, see [24]. We must prove that Γ is a geodesic on $\mathcal{S}$.

If Γ does not touch the boundary of $\mathcal{S}'$ then it is also a geodesic on $\mathcal{S}$. Suppose now that the point $P \in \Gamma$ belongs to the boundary of $\mathcal{S}'$. If we choose an orientation of Γ such that the angle θ_l of Γ at P is the one in the interior of $\mathcal{S}'$, then θ_l must be greater than or equal to π, because boundary points are treated as hyperbolic vertices. This means that the angle θ_l of $\partial\mathcal{C}$ at P is also greater than or equal to π, but as $\mathcal{C}$ is a convex set then, looking to $\partial\mathcal{C}$ as a curve on $\mathcal{S}$, the angle θ_r of $\partial\mathcal{C}$ at P must also be greater than or equal to π. Hence, P is a hyperbolic or euclidean point on $\mathcal{S}$ and the geodesic curvature $\kappa_s(P)$ of Γ at P is also zero when looking to Γ as a curve on $\mathcal{S}$. This means that Γ is a geodesic on $\mathcal{S}$. □

Remark 4.6 The converse of proposition 4.2 does not hold. In figure 4.2 we show a non-convex set, contained in a plane with a hole, where every pair of points can be joined by a shortest geodesic.

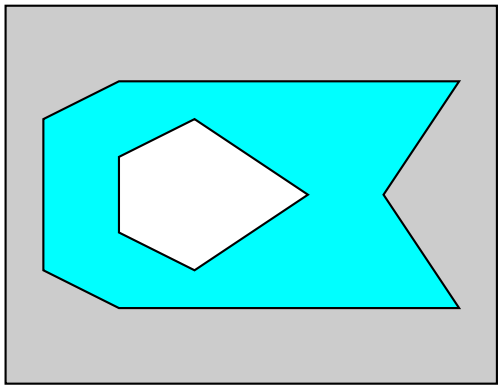


Figure 4.2: A set in plane with a hole. It contains a shortest geodesic between any pair of points although it is not convex.

4.3 Convex Hull

We want to define convex hull as the intersection of convex sets. To do that, we must study if convexity is preserved by this set operation. Unfortunately, this is not true; for example, the intersection of two meridians in a sphere results in a two-point set which is not even connected.

Although intersection does not preserve convexity, we still can use it to define convex hull. The following lemma states that each connected component of the intersection is a convex set:

Lemma 4.3 *The intersection of two convex sets $\mathcal{C}_1$ and $\mathcal{C}_2$ is a collection of convex sets.*

Proof: We know that the intersection of two convex sets may have more than one connected component. Let $\mathcal{C}$ be one of such connected components; we must prove that it is a convex set.

The boundary of $\mathcal{C}$ is formed by a sequence of curves belonging to the boundaries of $\mathcal{C}_1$ and $\mathcal{C}_2$. So, if we prove that κ_s remains smaller than or equal to zero at the intersection of the boundaries of $\mathcal{C}_1$ and $\mathcal{C}_2$, then we are done. But this is true because the left angle $\theta_l(\partial\mathcal{C}(P))$ at those points is smaller that both $\theta_l(\partial\mathcal{C}_1(P))$ and $\theta_l(\partial\mathcal{C}_2(P))$, and thus the geodesic curvature of $\partial\mathcal{C}$ at P is also negative:

$$\kappa_s(P) = \frac{2\pi}{\theta}\left(\frac{\theta}{2} - \theta_r\right) = \frac{2\pi}{\theta}\left(\frac{\theta_r + \theta_l}{2} - \theta_r\right) = \frac{2\pi}{\theta}\left(\frac{\theta_l - \theta_r}{2}\right) \leq 0.$$

The above formula holds everywhere, except at some hyperbolic vertices where θ_l is greater than π, but in those cases $\kappa_s(P) = 0$. Note that if P coincides with a spherical mesh vertex, then θ_l is necessary smaller than θ_r. □

After lemma 4.3 we can define the convex hull of curves on manifold triangulations:

Definition 4.3 *Let Γ be a curve defined on a surface $\mathcal{S}$. The convex hull $\overline{\Gamma}$ of Γ is the intersection of all convex sets containing Γ.*

$$\overline{\Gamma} = \bigcup_{\Gamma \subset \mathcal{C}_i} \mathcal{C}_i.$$

Remark 4.7 Note that we cannot extend the concept of convex hull in manifolds to arbitrary sets. For instance, the convex hull of two poles of a sphere would be the two points, which is not a connected set. However, this concept is well defined for curves, for which the convex hull is always a convex connected set.

The following proposition will be useful in section 4.5.

Proposition 4.4 *Given a simple polygonal curve Γ, joining two different points A and B, the area of its convex hull $\mathcal{A}(\overline{\Gamma})$ is equal to zero if and only if Γ is a simple shortest geodesic.*

Proof: If Γ is a simple shortest geodesic, then it is a convex set and consequently $\overline{\Gamma} = \Gamma$, so $\mathcal{A}(\overline{\Gamma}) = 0$.

Now assume that $\mathcal{A}(\overline{\Gamma}) = 0$, and suppose it is not a shortest geodesic. There exists an interior node $P_i \in \Gamma$ where $\kappa_s(P) \neq 0$. Hence, the set bounded by the geodesic joining P_{i-1} and P_{i+1}, and the segment of Γ between P_{i-1} and P_{i+1} belongs to $\overline{\Gamma}$. But the area of that set, which is a subset of $\overline{\Gamma}$, is not zero, which is a contradiction. □

After defining convex hull, it is interesting to know how its shape depends on the curve. Next lemma answer this question for polygonal curves.

Lemma 4.5 *The boundary of the convex hull $\overline{\Gamma}$ of a polygonal curve Γ is composed of a sequence of geodesic segments whose extremes are nodes of Γ.*

Proof: We know that $\kappa_s(\partial\overline{\Gamma}(.)) \leq 0$, suppose that $\kappa_s(P) < 0$ at a point $P \in \partial\overline{\Gamma}$. If $P \notin \Gamma$, then we can join two points of its neighboring sides in $\partial\overline{\Gamma}$ by a geodesic segment Γ_0 not crossed by Γ. Substituing the portion of $\partial\overline{\Gamma}$ between those points by Γ_0 we obtain a convex set $\mathcal{C}_0 \subset \overline{\Gamma}$ containing Γ, but then $\overline{\Gamma}$ would not be the convex hull of Γ. □

If the boundary $\partial\mathcal{C}$ of a convex set is a polygonal curve, we can distinguish two type of nodes in it, those where the geodesic curvature κ_s is zero, and those with non-null geodesic curvature. From now on, we refer to the last ones as vertices. Doing this, we can look at $\partial\mathcal{C}$ as a polygon whose sides are geodesics.

4.4 On the Metric of Discrete Surfaces

Until now, the definition of the length functional was the only metric property of triangulations needed for our exposition. In next section we will need other metric properties as distance and limit.

Given a connected discrete surface, as defined in chapter 1, a distance between its points can be defined.

Definition 4.4 *The* distance *between two points P and Q on a connected discrete surface $\mathcal{S}$ is defined as the length of an optimal shortest arc joining P and Q.*

$$\rho(P, Q) = \inf_{\Gamma_{PQ} \subset \mathcal{S}} \{L(\Gamma_{PQ})\}$$

As stated in lemma 2.2, if $\mathcal{S}$ is connected there always exists a shortest arc joining P and Q. Hence the distance ρ is well defined. The surface $\mathcal{S}$, provided with ρ, becomes a metric space [1, 2]. A metric like this one, where the distance between two points is the same as the length of the shortest arcs joining them, is called *intrinsic*. Note that on the interior of each face the metric of $\mathcal{S}$ coincides with the euclidean metric and then the length of curves, as defined in section 1.1.3, is the same as the length of curves on the intrinsic metric of $\mathcal{S}$.

A metric space M is complete if every Cauchy sequence converges to a point of M. For a space provided with an intrinsic metric we can give an equivalent definition of completeness, see [1].

Definition 4.5 *A metric of a space M is called* complete *if the Weierstrass theorem holds for M, i.e, if each infinite bounded set in M has an accumulation point.*

The following lemma is rather obvious and is probably proved somewhere else. However, we have not seen it proved and will give a proof here.

Lemma 4.6 *A discrete surface $\mathcal{S}$, provided with its intrinsic metric, is a complete metric space.*

Proof: From definition 4.5, it is enough to prove that every infinite set has an accumulation point.[1] Suppose that the set $A \in \mathcal{S}$ is infinite. Since $\mathcal{S}$ has a finite number of faces, there is a face $f \in \mathcal{S}$ which has infinite points of A. As in f the euclidean and intrinsic metric coincide, A has an accumulation point on f, and so it has an accumulation point on $\mathcal{S}$. □

The following proposition will be useful in next section.

Proposition 4.7 *Let $\{\mathcal{C}_n\}_{n=0}^{\infty}$ be a sequence of closed sets in a discrete surface $\mathcal{S}$ such that $\mathcal{C}_{n+1} \subset \mathcal{C}_n$ and $\mathcal{C}_n \neq \emptyset$. There exists a closet set $\mathcal{C} \neq \emptyset$ such that $\mathcal{C} = \bigcap_{n=0}^{\infty} \mathcal{C}_n$ and for all $\varepsilon > 0$ there exists an $n \in \mathbb{N}$ such that if $N \geq n$ then $\rho(P, \mathcal{C}) < \varepsilon$ for all $P \in \mathcal{C}_N$.*

Proof: The closed set $\mathcal{C}$ exists and is non-empty, see [21]. Lets prove the second part. Suppose that it does not hold, i.e, there is an $\varepsilon > 0$ such that $\forall n \in \mathbb{N}$ there exists a point $P_n \in \mathcal{C}_n$ such that $\rho(P_n, \mathcal{C}) \geq \varepsilon$. Consider the sequence $\{P_n\}$, from lemma 4.6 it has a subsequence $\{P_{k_n}\}$ convergent to a point P and $\rho(P, \mathcal{C}) \geq \varepsilon$. Given a $k \in \mathbb{N}$, $P_{k_n} \in \mathcal{C}_k$ for all $k_n \geq k$ and hence $P \in \mathcal{C}_k$, so $P \in \mathcal{C}$ and $\rho(P_n, \mathcal{C}) = 0$. But this is a contradiction with $\rho(P, \mathcal{C}) \geq \varepsilon$. □

4.5 Applications

The definition of convex hull of curves on triangulations help us to study the convergence of the geodesic algorithm presented in chapter 2.

In the rest of this chapter, $\{\Gamma_n\}_{n\in\mathbb{N}}$ denotes the sequence generated by algorithm 2.1. The following lemma will be used later:

Lemma 4.8 *In a triangulation $\mathcal{S}$, Γ_{n+1} belongs to the convex hull of Γ_n.*

Proof: At each vertex correction, for example at P_{nj}, the sequence $P_{n,j-1}P_{nj}P_{n,j+1}$ goes to the geodesic segment joining the points $P_{n,j-1}$ and $P_{n,j+1}$, which is entirely contained in the convex hull of the current curve Γ_n. So, each vertex of Γ_{n+1} belongs to the convex hull of Γ_n and so does Γ_{n+1}. □

Proposition 4.9 *If every Γ_n is simple, then the sequence converges to a geodesic line.*

Proof: Let $\mathcal{C}_n$ be the convex hull of Γ_n. Consider the set

$$\mathcal{C} = \bigcap_{n=0}^{\infty} \mathcal{C}_n.$$

From lemma 4.8 we know that $\mathcal{C}_n \bigcap \mathcal{C}_{n+1} = \mathcal{C}_{n+1}$. From proposition 4.7 we know that $\mathcal{C}$ exists and is closed. Each $\mathcal{C}_n$ is convex and so does $\mathcal{C}$.

We must prove that $\mathcal{A}(\mathcal{C}) = 0$ since, by proposition 4.4, only simple geodesic lines have convex hull with area equal to zero.

By proposition 4.7, for an arbitrary $\varepsilon > 0$ we can find a natural number n such that all the vertices in the boundary $\partial\mathcal{C}_n$ of $\mathcal{C}_n$ are at distance smaller than ε from the boundary $\partial\mathcal{C}$ of $\mathcal{C}$.

Suppose that $\mathcal{A}(\mathcal{C}) \neq 0$. Each vertex of $\partial\mathcal{C}_n$ is also a node of Γ_n, see lemma 4.5. There is an n large enough such that the vertices of $\partial\mathcal{C}_n$ are very close to $\partial\mathcal{C}$ and such that for each node $P_{nj} \in \Gamma_n \cap \partial\mathcal{C}_n$ the geodesic line joining its neighbors $P_{n,j-1}$ and $P_{n,j+1}$ in Γ_n intersects $\mathcal{C}$. When we compute the

[1] Since we are considering the case of discrete surfaces, which are bounded, every infinite set on them is also bounded.

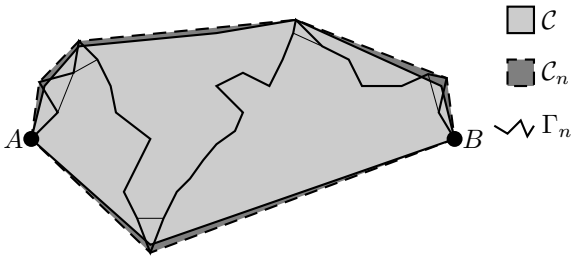


Figure 4.3: The convex hull of Γ_n is very close to C.

next curve Γ_{n+1}, it will pass through $\mathcal{C}$, and as P_{in} belongs to the boundary of the convex hull of Γ_n, there will be points of $\mathcal{C}$ outside of $\mathcal{C}_{n+1}$, but this is impossible since $\forall n\ \mathcal{C} \subset \mathcal{C}_n \ \Rightarrow \mathcal{C} \subset \mathcal{C}_{n+1}$. As a consequence $\mathcal{A}(\mathcal{C})$ must be zero and hence $\Gamma_n \underset{n\to\infty}{\longrightarrow} \Gamma$ which is a geodesic. □

Proposition 4.9 looks very restrictive, since we require that every curve Γ_n be simple. However, this seems to be the general case. When we look for an initial approximation, using FMM, the resulting curve Γ_0 is certainly simple. After that, each correction is performed in a neighborhood of a curve node and it us unlikely that self-intersection will occur.

To finish this section, we present another application of convex hulls:

Proposition 4.10 *Geodesic Bézier curves satisfy the convex hull property. This is, a geodesic Bézier curve is completely inside the convex hull of its control polygon.*

Proof: This proposition is a direct consequence of lemma 4.8. □

4.6 Non-Polygonal Curves

Up to now, all the material in this chapter was mainly dedicated to polygonal curves. In fact, as far as this work is concerned, polygonal curves are enough. Nevertheless, we can extend our study about convexity to sets with non-polygonal boundaries. We can also extend it to smooth manifolds. For the sake of completeness, we consider them here.

Consider a piecewise smooth curve Γ on a triangulation $\mathcal{S}^2$. At points where the curve is smooth, it has neighborhoods which are isometric to a piece of plane, so we define the geodesic curvature κ_s of Γ at those points as the corresponding curvature in the unfolding. At non-smooth points, we define κ_s as the curvature corresponding to the polygonal line defined by the left and right tangent on these points, when it exists. Otherwise κ_s is not defined.

On smooth manifolds, the geodesic curvature is always defined for smooth curves. Consider a piecewise smooth curve γ, which is simple and closed. In points where γ is not smooth, we can look at the corresponding left and right tangent lines in the plane tangent to the manifold. If the angle[3] between them is smaller than π, and the geodesic curvature at smooth points does not change its sign, then the set bounded by γ is convex.

[2]Note that polygonal curves are special cases of piecewise smooth curves.
[3]This angle is measured in the same direction of the curve.

5 Conclusion

We have presented an iterative algorithm to compute a shortest geodesic between two points over a discrete surface. At each step it computes a new curve with smaller length. This is done by reducing locally the curve length at each vertex. It explores the fact that the intersection of a mesh face with a shortest geodesic is a line segment, and hence its vertices lie on mesh vertices or edges. The proposed iterative process allows also to improve an approximation given by any other (non-exact) algorithm. As far as the author knows, there is not other algorithm which improves discrete geodesic approximations. We have also given an strategy to speed up the convergence, it is specially useful when part of the initial curve is close to be a geodesic as in the case of geodesic Bézier curves.

Using the geodesic algorithm, we defined geodesic Bézier curves, which are a generalization of Euclidean Bézier curves to manifold triangulations, and studied some properties of them. They have the advantage of being defined intrinsically, which makes them independent of any parameterization. We have shown how to use them to define pieces or regions of a surface, allowing trimming, local texture mapping, and region coloring. Fast user interaction joined with the possibility of constructing C^0 and C^1 splines make of them a powerful tool for free-form modeling on manifold triangulations.

Convex sets on discrete surfaces were defined by imitating the property of plane convex sets of having boundary curves that do not change the sign of the curvature. There were studied some properties of convex sets on triangulation and it was defined convex hull of curves in this context. As applications we get the convex hull property of geodesic Bézier curves and studied the convergence of the geodesic algorithm.

5.1 Further Research

Although using a heap in the geodesic algorithm diminishes the amount of node corrections, we have not great improvement in performance because at each correction the neighboring nodes must be updated in the heap. It is interesting to study other correction strategies as the multiple choice approach, where we randomly select few nodes and correct the one with largest error. It is very important to study the order of convergence of the geodesic algorithm. It is also useful to compare the result with different triangulations of simple surfaces as the cylinder, where it is trivial to compute the geodesics.

An interesting application of the geodesic algorithm would be to parameterize a triangulation using the exponential map, what can lead to well-shaped charts.

There remain some theoretical issues associated with geodesic Bézier curves; it will be very interesting to see which other properties of classical Bézier curves hold for the new curves and also which concepts can be generalized to the geometry of manifold triangulations. For example, it is not clear how to define the control polygons if we want C^2 continuity or higher. The continuity of the curves at mesh vertices has to be studied. One should also investigate whether there is something equivalent to affine invariance

of Bézier curves in the case of geodesic Bézier curves. It would be interesting to compare the properties of the limit curves defined by the de Casteljau's subdivision algorithm with those of the well-known subdivision schemes for curves and surfaces.

There are some works about geodesic computation in geometries other than manifold triangulations. So, we can define geodesic Bézier curves for point clouds [23], for Riemannian manifolds [19], and for smooth surfaces [17, 16]. The next step is to study how to handle user interaction in a fast way and what properties of classical Bézier curves are inherited by geodesic Bézier curves on those geometries. A good point to start could be subdivision surfaces where the extension of the ideas in this thesis seems to be straightforward, with the nice property that user interaction could be handled at low resolution, making it faster.

The extension of these results to other subdivision schemes, as well as the definition of interpolant splines will certainly result in a design toolkit with wider capabilities.

Bibliography

[1] A. D. Aleksandrov. *A. D. Aleksandrov selected works. Intrinsic Geometry of Convex Surfaces*. Chapman & Hall/CRC, 2006.

[2] A. D. Aleksandrov and V. A. Zalgaller. *Intrinsic Geometry of Surfaces*, volume 15 of *Translation of Mathematical Monographs*. AMS, 1967.

[3] H. Biermann, I. Martin, F. Bernardini, and D. Zorin. Cut-and-paste editing of multiresolution surfaces. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 312–321, New York, NY, USA, 2002. ACM Press.

[4] H. Biermann, I. M. Martin, D. Zorin, and F. Bernardini. Sharp features on multiresolution subdivision surfaces. *Graph. Models*, 64(2):61–77, 2002.

[5] J. Chen and Y. Han. Shortest paths on a polyhedron. In *Proceedings of 6th Annu. ACM Sympos. Comput. Geom*, pages 360–369, 1990.

[6] L. C. G. Coelho, M. Gattass, and L. H. de Figueiredo. Intersecting and trimming parametric meshes on finite-element shells. *International Journal for Numerical Methods in Engineering*, 47(4):777–800, 2000.

[7] E. Cohen, R. F. Riesenfeld, and G. Elber. *Geometric Modeling with Splines: An introduction*. A K Peters, Ltd., 63 South Avenue, Natick, MA 01760, 2001.

[8] P. Crouch, G. Kun, and F. S. Leite. The de Casteljau algorithm on Lie groups and spheres. *Journal of Dynamical and Control Systems*, 5(3):397–429, July 1999.

[9] P. de Casteljau. Outillage Méthodes Calcul. Internes Dokument P2108, SA André Citroën, Paris, Feb. 1959.

[10] M. P. Do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, 1976.

[11] G. Farin. *Curves and surfaces for CAGD: a practical guide*. Morgan Kaufmann Publishers Inc., San Francisco, 2002.

[12] M. S. Floater and K. Hormann. Parameterization of triangulations and unorganized points. In A. Iske, E. Quak, and M. S. Floater, editors, *Tutorials on Multiresolution in Geometric Modelling*, Mathematics and Visualization, pages 287–316. Springer, 2002.

[13] M. S. Floater and K. Hormann. *Surface Parameterization: a Tutorial and Survey*, pages 157–186. Springer-Verlag, Heidelberg, 2005.

[14] M. Hofer and H. Pottmann. Energy-minimizing splines in manifolds. *ACM Trans. Graph.*, 23(3):284–293, 2004.

[15] S. Kapoor. Efficient computation of geodesic shortest paths. In *Proceedings of 31st Annu. ACM Sympos. Theory Comput.*, pages 770–779, 1999.

[16] E. Kasap, M. Yapici, and F. T. Akyildiz. A numerical study for computation of geodesic curves. *Applied Mathematics and Computation*, 171(2):1206–1213, 2005.

[17] R. Kimmel and G. Sapiro. Shortening three-dimensional curves via two-dimensional flows. *Computers and Mathematics with Applications*, 29(3):49–62, 1995.

[18] R. Kimmel and J. Sethian. Computing geodesic paths on manifolds. In *Proceedings of the National Academy of Sciences of the USA*, 95(15):8431–8435, July 1998.

[19] E. Klassen, A. Srivastava, W. Mio, and S. Joshi. Analysis of planar shapes using geodesic paths on shape manifolds. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(3):372–384, 2004.

[20] S. Krishnan and D. Manocha. An efficient surface intersection algorithm based on lower dimensional formulation. *ACM Trans. on Computer Graphics*, 16(1):74–106, 1997.

[21] E. Lages. *Curso de análise*, volume 2. Projeto Euclides, 5 edition, 1999.

[22] N. Litke, A. Levin, and P. Schröder. Trimming for subdivision surfaces. *Computer Aided Geometric Design*, 18(5):463–481, June 2001.

[23] F. Mémoli and G. Sapiro. Distance functions and geodesics on submanifolds of $\mathbb{R}^d$ and point clouds. *SIAM Journal on Applied Mathematics*, 65(4):1227–1260, 2005.

[24] J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou. The discrete geodesic problem. *SIAM J. Comput.*, 16:647–668, 1987.

[25] F. C. Park and B. Ravani. Bezier curves on Riemannian manifolds and Lie groups with kinematic applications. *ASME Journal of Mechanical Design*, 117:36–40, 1995.

[26] K. Polthier and M. Schmies. Straightest geodesics on polyhedral surfaces. In H.-C. Hege and K. Polthier, editors, *Visualization and Mathematics*, pages 135–150. Springer Verlag, Heidelberg, 1998.

[27] K. Polthier and M. Schmies. Geodesic flow on polyhedral surfaces. In *Data Visualization, Proceedings of Eurographics Workshop on Scientific Visualization*, Vienna, 1999. Springer Verlag.

[28] H. Pottmann and M. Hofer. A variational aproach to spline curves on surfaces. *Computer Aided Geometric Design*, 22(7):693–709, October 2005.

[29] R. C. Rodriguez, F. S. Leite, and J. Jacubiak. A new geometric algorithm to generate smooth interpolating curves on riemannian manifolds. *LMS Journal of Computation and Mathematics*, (8):251–266, 2005.

[30] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. In *Proceedings of the National Academy of Sciences of the USA*, 93(4):1591–1595, February 1996.

[31] O. Sifri, A. Sheffer, and C. Gotsman. Geodesic-based surface remeshing. In *Proceedings of 12th International Meshing Roundtable*, 2003.

[32] V. Surazhsky, T. Surazhsky, D. Kirsanov, S. Gortler, and H. Hoppe. Fast exact and approximate geodesics on meshes. In *Proceedings of ACM SIGGRAPH 2005*, pages 553–560, 2005.

[33] J. Wallner. Smoothness analysis of subdivision schemes by proximity. *Constr. Approx.*, 24(3):289–318, 2006.

[34] J. Wallner and N. Dyn. Convergence and C^1 analysis of subdivision schemes on manifolds by proximity. *Comput. Aided Geom. Design*, 22(7):593–622, 2005.

[35] J. Wallner and H. Pottmann. Intrinsic subdivision with smooth limits for graphics and animation. *ACM Trans. Graphics*, 25(2):356–374, 2006.

[36] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. Shum. Mesh editing with poisson-based gradient field manipulation. *ACM Trans. Graph.*, 23(3):644–651, 2004.