Geodesic Bézier Curves: a Tool for Modeling on Triangulations

Dimas Martínez Morera Paulo Cezar Carvalho Luiz Velho
Instituto Nacional de Matemática Pura e Aplicada - IMPA
Estrada Dona Castorina, 110, Jardim Botânico, Rio de Janeiro, RJ. 22460-320

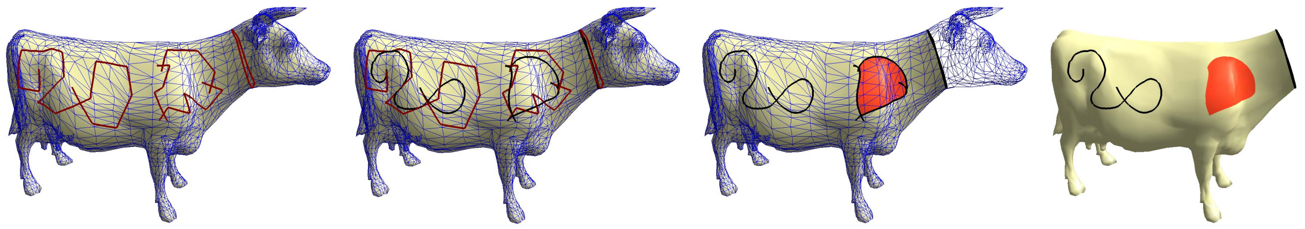

Figure 1. Modeling on the surface of a cow. From left: the control polygons of some curves and a C^1 spline, the corresponding curves, a region is filled and other is trimmed, final result.

Abstract

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.*

Keywords: *Geodesic Bézier Curve, Discrete Geodesic, de Casteljau Algorithm, Spline Curves, Free-Form Design.*

1. Introduction

Designing free-form curves is a basic operation in Geometric Modeling. Doing so in Euclidean space is a widely studied problem; see [5, 8] 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 paper 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 "Geodesic Bézier Curves".

1.1. Overview

We begin in section 2 with a summary of related work. In section 3 we give a brief overview of the state of the art of Discrete Geodesic computation and present a simple description of the algorithm given by [18], which will be used in sections 4 and 5. After summarizing classical Bézier curves theory, we define in section 4 the new class of curves and compare them to the classical ones. A study of their use in modeling operations is done in section 5. Section 6 describes the construction of piecewise Bézier spline curves on triangulations. Finally in section 7 we give concluding remarks and indicate potential further research.

1.2. Notation and Preliminary Definitions

A polygonal line Γ on a triangle mesh $\mathcal{S}$ is defined as a sequence of nodes $\{\Gamma_0, \Gamma_1, \ldots, \Gamma_n\} \subset \mathcal{S}$ such that every line segment $\overline{\Gamma_i \Gamma_{i+i}}$ is also contained in $\mathcal{S}$. We refer to polygonal vertices as nodes, in order to differentiate them from mesh vertices.

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 geodesic 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. However, geodesic Bézier curves in practice are well behaved when they pass through mesh vertices. Furthermore, we are currently investigating the theoretical aspects of curve continuity at mesh vertices.

2. Related Work

The de Casteljau algorithm has been adapted to Riemannian manifolds using geodesic interpolation [21]. However, it has been applied only to some surfaces where geodesic computation is relatively easy, such as spheres or Lie groups; see [6, 24] 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 [21], but we consider manifolds triangulations.

Another generalization of cubic splines to smooth manifolds is given in [9, 23]. 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 [15, 4]. An approach for subdivision surfaces is to modify the original (coarse) mesh in order to obtain a trimmed limit surface [2, 16]. Our trimming operations are done directly on the mesh and no parameterization is needed.

Recent works [28, 27, 26] define a general framework for curve subdivision schemes. Geodesic Bézier 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 paper 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. Geodesic Curves

The problem of computing locally shortest geodesic paths on discrete geometries, particularly meshes, has been addressed in many works [1, 3, 10, 13, 20], and it is still subject of active research [18, 25]. Most algorithms use the so-called Continuous Dijkstra Technique. The algorithm proposed in [18] adopts an iterative curve correction strategy that we believe is suitable to our curve design algorithm, with the goal of reaching responsive user interaction without increasing storage space. We will go back to this subject in section 4.

3.1. Iterative Curve Correction Algorithm

In this section we summarize the Iterative Curve Correction Algorithm; for details see [18]. Geodesic computation is performed in two steps. In the first step, an initial polygonal curve, joining two points in the mesh, is computed using a front propagation strategy. In the second step, all the nodes of the initial curve are put in a priority queue; then the node with largest error is corrected and the error at neighboring nodes is updated. This process is repeated until a small error is attained.

Nodes are constrained to lie on mesh edges since a geodesic must coincide with a line segment in the interior of each face, and the extremes of the curve are added as new vertices to the mesh. Errors at curve nodes are computed based on discrete geodesic curvature [22]. A node position is corrected by unfolding a subset of the faces adjacent to it and moving it to the line joining its neighboring nodes in the unfolded part of the mesh.

Since our curves are allowed to pass through the interior of a face – not necessarily as a line segment – we would need to add new vertices to the mesh any time we subdivide a control polygon. However, a careful implementation would allow geodesic nodes to lie in the interior of mesh faces, leaving the underlying mesh intact.

4. 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.

4.1. 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:

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

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 [5, 8]; figure 2 shows an example of a Bézier curve of degree 3.

The de Casteljau Algorithm [7] provides a geometric procedure to evaluate a Bézier curve at any parameter $u \in [0, 1]$, using repeated linear interpolation:

Algorithm 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. $P(u) = P_n^{[n]}$
```

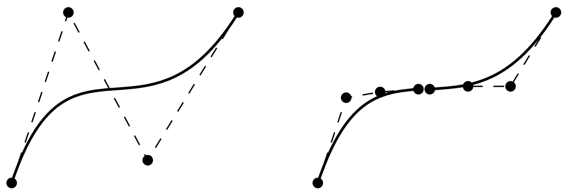

Figure 2. A subdivision step of a control polygon and its Bézier curve.

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

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 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.
        $P([0,u])$ =bezier($P_0^{[0]}, P_1^{[1]}, \ldots, P_n^{[n]}$)
        $P([u,1])$ =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 2 shows a subdivision step of the control polygon of a degree 3 Bézier curve.

Algorithm 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".

4.2. Bézier Curves on Triangulations

Geodesic 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 2 of section 4.1. 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

Algorithm 3 : *Interpolation on 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 $\gamma$ such that
              $d_\gamma(Q_1, Q) = u\, d_\gamma(Q_1, Q_2)$
```

step on algorithm 1. The equivalent to linear interpolation in the geometry of the surface is the interpolation along geodesic lines. Algorithm 3 describes the interpolation step in the case of manifold triangulations. In this algorithm, $\mathrm{d}_\gamma(A, B)$ 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 shows some geodesic Bézier curves along with their control polygons.

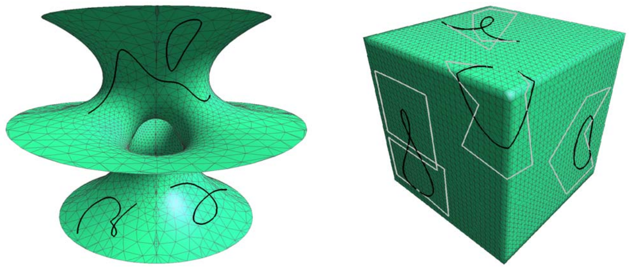

Figure 3. Some geodesic Bézier curves. Control polygons are also shown in the Cube model.

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. 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. 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 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 u. In practice, selecting a fixed value of u gives us a subdivision curve. In this paper we have chosen $u = 0.5$ and therefore we have a midpoint subdivision scheme. All figures of this paper were generated using this scheme.

Geodesic algorithm selection

There are several algorithms to compute geodesics (see section 3) and any of them could be used both to compute geodesic Bézier curves and to perform user interaction. We chose the algorithm of [18] 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 [25] 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.

4.3. Properties of Geodesic Bézier Curves

Geodesic Bézier curves share some properties with classical ones. The proofs of the following propositions can be found in [17].

Proposition 1 *Geodesic Bézier curves interpolate P_0 and P_n and are tangent to the control polygon at these points.*

Each interior point of an edge has a neighborhood which is isometric to an open disc in the plane. By means of this isometry, we can analyze the behavior of a curve when passing through the interior of mesh edges. If the curve is smooth in the plane, it will have smooth appearance in the mesh. The following proposition address this property of geodesic Bézier curves.

Proposition 2 *A geodesic Bézier curve has (at least) C^1 continuity when intersecting an edge of the mesh.*

Proposition 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.*

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 part of our current research.

The Convex Hull property of Bézier curves has a huge importance in modeling. The adaptive version of de Casteljau's algorithm relies on this property. It is not trivial to give a proper definition of *convex set* in a curved geometry. However, we can find in [17] definitions of convex set and convex hull that are appropriated for the study of our curves.

The following propositions guarantee the correctness of the adaptive version of geodesic Bézier curves.

Proposition 4 *Geodesic Bézier curves satisfy the convex hull property. This is, a geodesic Bézier curves is contained in the convex hull of its control polygon.*

Proposition 5 *Given a simple polygonal curve Γ, joining two different points A and B on $\mathcal{S}$, the area of its convex hull $\mathcal{A}(\overline{\Gamma})$ is equal to zero if and only if Γ is a simple shortest geodesic.*

5. 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.

5.1. 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 the corresponding extreme in the old curve. Hence we use, as initial approximation for the algorithm described in section 3, 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 4 shows three different positions for the middle node during user interaction with a third order curve.

5.2. 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 4 describes how to identify the faces in the region $\mathcal{R}$ that contains the point P.

Algorithm 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 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 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 5 (left). Once we have identified the set $S_{\mathcal{R}}$ of faces cutting $\mathcal{R}$, we

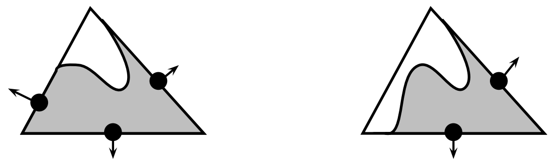

Figure 6. Propagation directions. Arrows indicate by what edges can the wave be propagated. Bullets indicate what portion of the edges belong to $\mathcal{R}$ (shadowed region).

must decide which part of the boundary faces belongs to $\mathcal{R}$. To do that, during propagation we mark each portion of an

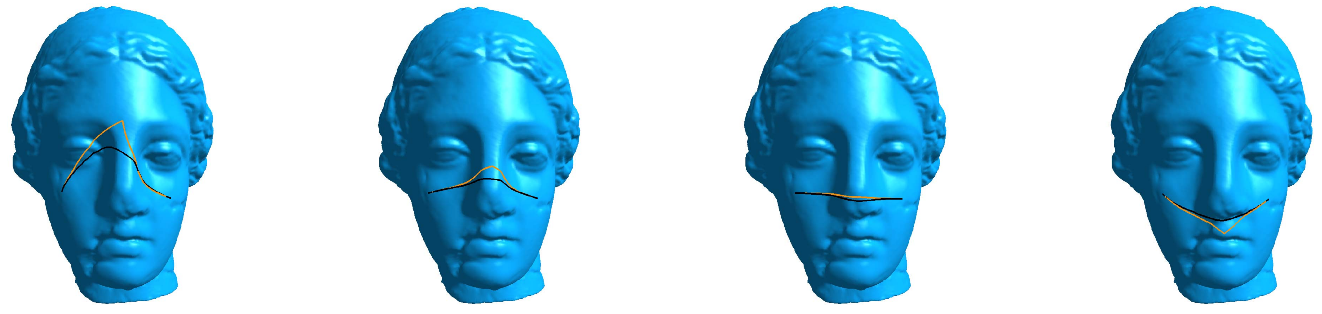
Figure 4. Igea model: Four different positions for the middle control point in a curve with three control points.

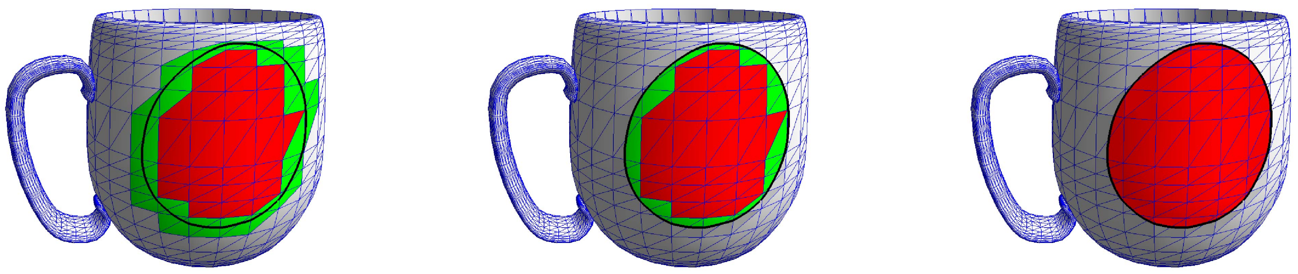
Figure 5. Region finding stages. Left: set $S_{\mathcal{R}}$ with boundary faces highlighted. Middle and Right: the region $\mathcal{R}$ after eliminating the part of boundary faces not belonging to it.

edge intersecting $\mathcal{R}$, see figure 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 7). For texture mapping or trimming it is not sufficient to identify 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 8 shows some regions filled or trimmed in the Cube and the Bunny models.

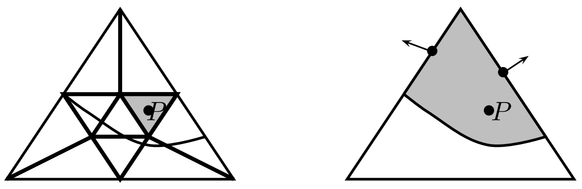

Figure 7. Locating seed point

6. 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 of low degree. We want to compute piecewise 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

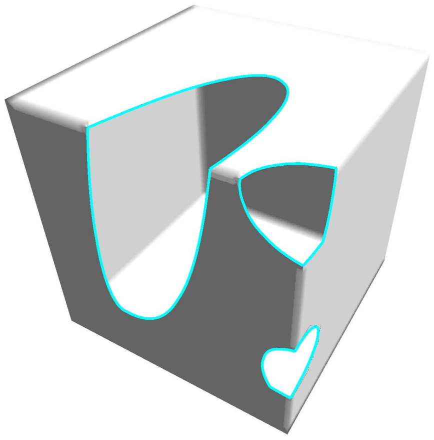
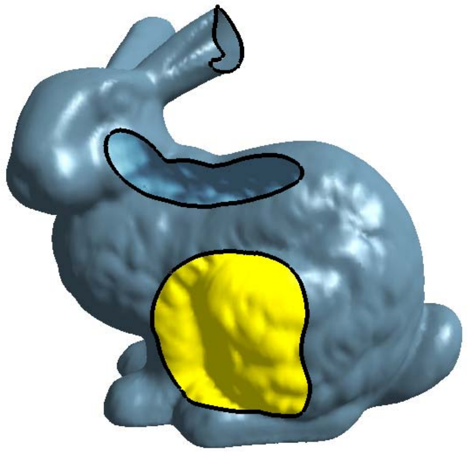

Figure 8. Filled and trimmed regions. Up: Trimmed Cube. Down: Stanford's bunny with two trimmed regions and a filled one.

to continue the geodesic line between P^i_{m-1} and P^i_m. 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 [22] 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 [22] 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 4.3 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^i_{m-1} modifies the position of P^{i+1}_1 and vice versa. In the last case,

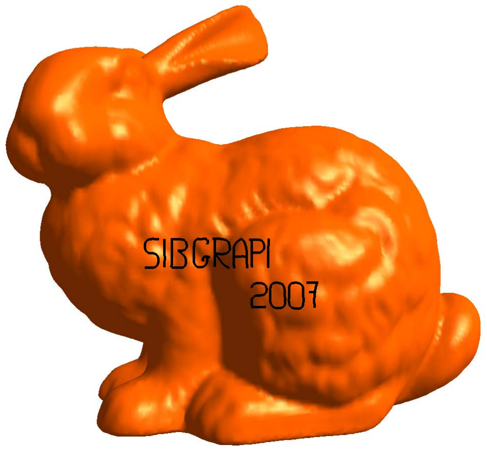

Figure 9. C^0 and C^1 splines on the surface of the bunny.

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^i_{m-1} and P^{i+1}_1. Figure 9 show the use of C^1 splines to write in the surface of the Stanford's bunny model. The middle curve in figure 1 is a C^1 spline, composed by 8 Bézier segments.

7. Conclusion

We have 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 geodesically, 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.

7.1. Further Research

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. Is there something equivalent to affine invariance of Bézier curves in the case of geodesic Bézier curves?

There are some works about geodesic computation in geometries other than manifold triangulations. So, we can define geodesic Bézier curves for point clouds [19], for Riemannian manifolds [14], and for smooth surfaces [12, 11]. 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 paper seems to be straightforward, with the nice property that user interaction could be handled at low resolution, making it faster.

8. Acknowledgements

This research has been developed in the VISGRAF Laboratory at IMPA. VISGRAF Laboratory is sponsored by CNPq, FAPERJ, FINEP and IBM Brazil. We would like to thank Uri Usher and Luiz Henrique de Figueiredo for their valuable sugestions after reading the original manuscript. Most of the 3D models used were taken from the Stanford 3D Scanning Repository (http://www-graphics.stanford.edu/data/3Dscanrep/) and the Cyberware Sample Models (http://www.cyberware.com/).

References

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

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

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

[4] 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.

[5] 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.

[6] 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.

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

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

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

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

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

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

[17] D. Martínez. *Geodesic-based Modeling on Manifold Triangulations*. PhD thesis, IMPA, Rio de Janeiro, Brazil, 2006.

[18] D. Martínez, L. Velho, and P. C. Carvalho. Computing geodesics on triangular meshes. *Computer and Graphics*, 29(5):667–675, October 2005.

[19] 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.

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

[21] 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.

[22] 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.

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

[24] 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.

[25] 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.

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

[27] 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.

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