Computing Geodesics on Triangular Meshes ⋆

Dimas Martínez [a,b,*], Luiz Velho [a], Paulo C. Carvalho [a]

[a] *IMPA–Instituto Nacional de Matemática Pura e Aplicada*
Estrada Dona Castorina, 110, 22460-320 Rio de Janeiro, RJ, Brasil

[b] *ICIMAF–Instituto de Cibernética, Matemática y Física*
Calle E, 309, Vedado, Ciudad de la Habana, Cuba

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.

Key words: shortest geodesic, manifold triangulation, curve evolution

1 Introduction

Geodesic curves are useful in many areas of science and engineering, such as robot motion planning, terrain navigation, surface parameterization [1], remeshing [2] and front propagation over surfaces [3]. 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 [4,5], 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

⋆ Based on "Geodesic Paths on Triangular Meshes", by D. Martínez, L. Velho and P. C. Carvalho which appeared in Proceedings of SIBGRAPI/SIACG 2004 (ISBN 0-7695-2227-0).

* Corresponding author.
Email addresses: `dimasmm@visgraf.impa.br` (Dimas Martínez), `lvelho@visgraf.impa.br` (Luiz Velho), `pcezar@visgraf.impa.br` (Paulo C. Carvalho).

a geodesic as a shortest path between two points on the surface is maybe the more widely used definition.

In this paper we are concerned with the problem of compute 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 [6] 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 [7] 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 himself [8] 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 [9] 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 [9,7,8,6,10] and the references therein.

In section 2 we do a quick review of the definitions for smooth and discrete surfaces. Section 3 presents the algorithm, which is the main contribution of this paper. We begin with an approximate path and use an iterative process to approach the true geodesic path. Finally in section 4 we show some experimental results and adapt our algorithm to the single source shortest path problem.

1.1 Preliminary notations and definitions

We will restrict the study of the discrete geodesics computation to manifold triangulations. An extension to non-manifold discrete surfaces is left to future work. We consider a discrete surface S as a finite set F of (triangular) faces such that:

(1) Any point $P \in S$ lies in at least one triangle $f \in F$.

(2) The intersection of two different triangles $g, h, \in F$ is either empty, or consists of a common vertex, or of a common edge.
(3) Each point $P \in S$ has a neighborhood which is either homeomorphic to a disc, or homeomorphic to a semi-disc.

We denote by a greek letter (γ, α, $\cdots$) a curve over a smooth surface, and use capital greek letters (Γ, Γ_i, $\cdots$) to denote curves over discrete surfaces.

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

2 Geodesic Curves

Geodesic curves generalize the concept of straight lines for smooth surfaces. Therefore, they have several "good" properties, discussed on section 2.1. 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 2.2, that depend on their proposed use. The rest of this section was mainly extracted from references [11,5].

2.1 Geodesic curves in 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 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 [11].

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 *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 proposition characterizes geodesic curves.

Proposition 3 *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 3 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 and B is 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.

2.2 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 happens to be very improbable. 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 3 hold.

[1] The geodesic curvature κ_g generalizes to surfaces the concept of curvature of a plane curve. See reference [11].

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 [4] as limit curves of geodesics on a family of converging smooth surfaces. He also defined discrete geodesics as critical points of the length functional over polyhedral surfaces; in other words, he 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 paper is to find a shortest geodesic joining two points over a triangular mesh.

Polthier and Schmies [5] defined *straightest geodesics* inspired in the characterization of smooth geodesics given by item 4 of proposition 3. They defined *discrete geodesic curvature* as a generalization 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 1), the discrete geodesic curvature is defined as

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

Choosing θ_l instead of θ_r 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.

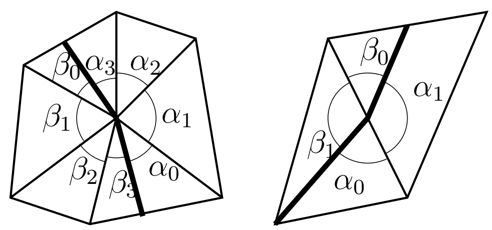


Fig. 1. Right and left angles (θ_r and θ_l resp.) in a curve. $\theta_r = \sum \alpha_i$ and $\theta_l = \sum \beta_i$.

Definition 4 *A mesh vertex is classified by 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 $2\pi - \theta < 0$.

The following proposition explores the difference between straightest and shortest geodesic. It was proved by Polthier and Schmies [5] and will be very useful in defining an strategy to compute a shortest geodesic.

Proposition 5 *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.

3 Geodesic Computation

The algorithm that we propose to compute a shortest geodesic Γ between A and B 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 1 and the next two subsections explain in detail how to perform them.

Algorithm 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 Γ_i $(i = 0, 1, \cdots)$ to reach a good approximation Γ_n of Γ*

3.1 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. Thus, in the following, A and B will be treated as vertices.

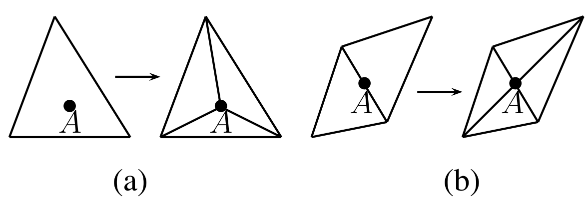
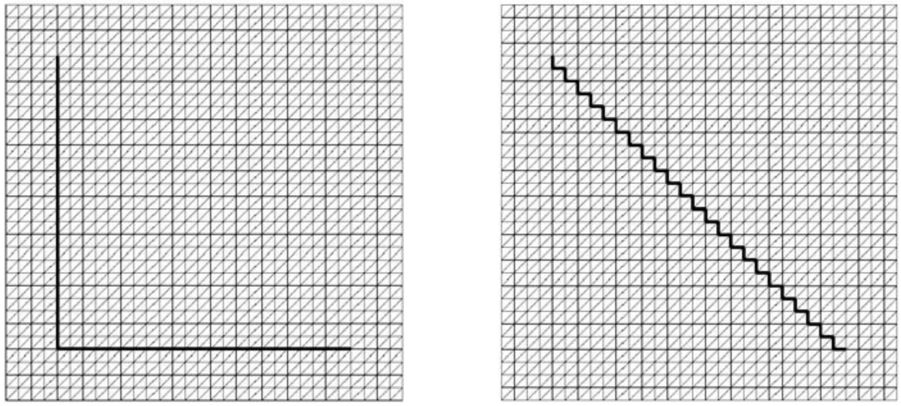


Fig. 2. Inserting the point A in S as a mesh vertex: it belongs to the interior of a face (a) and to an edge (b).

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 curve 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 3 we compare the results of using Dijkstra's Algorithm and FMM to compute Γ_0 in a regular plane triangulation.



(a) (b)

Fig. 3. 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 [8]. They solve the Eikonal equation

$$|\nabla T| = 1$$

where $T(P)$ is the (geodesic) distance from A to any point P on S (see references [8,12] for details). The efficiency of this process relies on the propagation of T over S maintaining a narrow band of vertices close to the front. Once T is computed for every vertex, they must solve the ordinary differential equation

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

to get the geodesic path $\chi(s)$. To integrate this equation, using Huen's method, T is approximated in the interior of a face by interpolating a second degree polynomial to the previously computed values of T 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. The correctness of this step is guaranteed since the distance T was defined increasingly from A. Moreover, the same argument permits us to stop FMM once $T(B)$ is computed. The remaining points (where T was not defined) will have $T(P) = \infty$.

Algorithm 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 $T(P)$ 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* $T(P_{i+1})$ *from A.*

Put P_{i+1} *in* Γ_0

$i = i + 1$

Even in the case where 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.

3.2 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 vertices, 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, Γ_1, Γ_2 and so on, to lie on edges or vertices of S.

We will use different procedures to correct the positions of the vertices 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.

3.2.1 Correction of a vertex in the interior of an edge.

Suppose the polygonal curve Γ_i is given by the sequence of vertices $B = P_{i0}, P_{i1}, \cdots, P_{in} = A$. For a vertex P_{ij} ($j \in \{1, 2, \cdots, n-1\}$) lying in the interior of an edge 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 E, and define the new $P_{i+1,j'}$ as the intersection point of E with the line joining $P_{i,j-1}$ and $P_{i,j+1}$ (see figure 4 (a)).

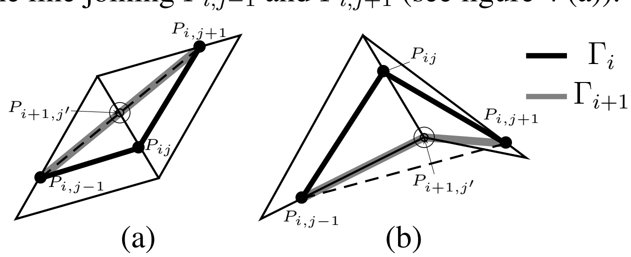


Fig. 4. Correcting the vertex position on an edge. Intersection inside the edge (a), and outside (b). Corrected polygonal vertices are marked with a ⊚.

Sometimes there is not an intersection point, as in figure 4 (b). In such cases we replace P_{ij} by the vertex of E which is closer to the intersection point between the line containing E and the line passing by $P_{i,j-1}$ and $P_{i,j+1}$.

In both cases the corrected vertex $P_{i+1,j'}$ gives the shortest curve passing by $P_{i,j-1}$ and $P_{i,j+1}$ inside the two triangles sharing edge E. In some cases (see section 3.2.2) it is necessary to replace vertex P_{ij} by more than a vertex when correcting Γ_i. Also, some vertices may be eliminated. These facts justify the notation $P_{i+1,j'}$ used above for the corrected vertex in curve Γ_{i+1}.

3.2.2 *Correction of a vertex which is also 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 $P_{i,j-1}$ 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 S; then, such union of triangular faces will be called S_k. For simplicity $P_{i,j-1}$ and $P_{i,j+1}$ are supposed to be on the boundary of S_k; otherwise one of them belongs to the interior of S_k and in that case we can eliminate it from Γ_i without any loss of information. In fact, this vertex elimination will result in a shortest curve (see figure 5).

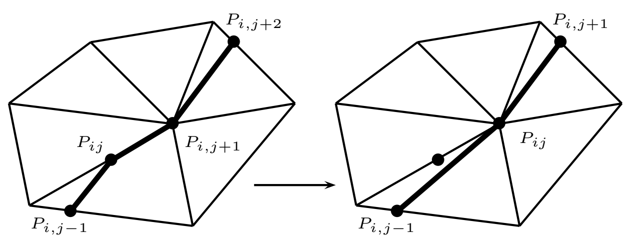


Fig. 5. Elimination of a vertex inside S_k.

We first classify vertex P_{ij} as in definition 4 by computing left and right angles θ_l and θ_r, and then we shorten the curve by taking into account proposition 5. If P_{ij} is euclidean then S_k can be isometrically unfolded to be part of a plane and we just have to join the images of $P_{i,j-1}$ and $P_{i,j+1}$ in the unfolding of S_k. If P_{ij} is spherical then no shortest curve may pass through it; in this case choose the part of S_k with smaller angle, flatten it up, and join $P_{i,j-1}$ 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 5 [5]). If one of the angles, say θ_r, is smaller than π then the geodesic must pass through the corresponding side of S_k; we then proceed to flatten it up an compute the line joining $P_{i,j-1}$ 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 S_k and we have to insert them in the polygonal curve in the correct order. Like in section 3.2.1, it could happen that the intersection point is outside of some edge (see figure 6); 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 vertex correction on the extreme of the corresponding edge, obtaining the shortest path between the neighbors of P_{ij} [2].

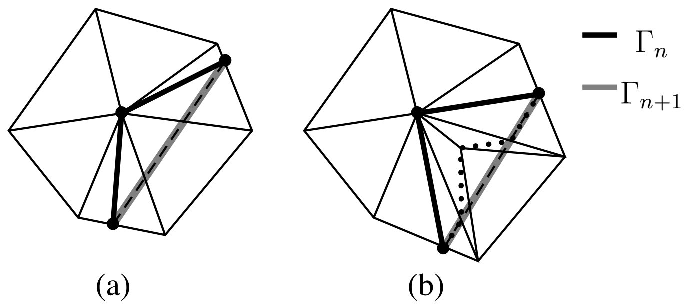


Fig. 6. Correcting a vertex coinciding with a mesh vertex. Convex star (a) and non-convex star (b). The resulting path before the additional correction in the non-convex vertex is shown as a dotted line.

3.2.3 Some remarks on path correction.

Algorithm 3 summarizes the path correction step. It is inspired in Polthier's straightest geodesics theory, more precisely in the characterization given in proposition 5 about the differences between shortest and straightest geodesics.

Algorithm 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 $j = 1, 2, \cdots, n-1$
    if P_{ij} *belongs to an edge*
      correct P_{ij} *using section 3.2.1*
    else
      correct P_{ij} *using section 3.2.2*

To get better path correction, and speed up the iteration convergence as a consequence, we use $P_{i+1,j'-1}$, the last corrected vertex, instead of $P_{i,j-1}$. We will get a better correction $P_{i+1,j'}$ since we use a vertex whose position was previously corrected. Besides, with this simple modification we are able to prove that our algorithm actually reduces the length of Γ_i at each step.

In section 3.2.2 we chose to trace the geodesic line in the side of S_k with smaller angle. This election was not arbitrary. At hyperbolic vertices, the geodesic can only be traced on the smaller angle's side since the other side cannot be flattened. On the other hand, at spherical vertices it is possible to flatten both sides, but the law of the cosines ensures that the shortest path is obtained in the side with smaller angle. Although the problem of selecting the right side of S_k to look for $P_{i+1,j'}$ seems not

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

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 S_k is part of a plane (otherwise we can flatten it isometrically), and consider the triangle formed by $P_{i,j-1},\ P_{ij}$ 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.

3.3 Implementation issues

3.3.1 Stop criterion.

An iterative process should always be controlled by a stop criterion, usually based on some error measure. Maybe 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 vertex based on proposition 5, and then define a curve error measure as the maximum vertex error. For vertices lying in the interior of mesh edges and vertices coinciding with euclidean mesh vertices, we define the error as the difference between left and right angles θ_l and θ_r. For vertices coinciding with spherical mesh vertices we define the error as a huge value, since no shortest geodesic can pass through it. For vertices 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.

3.3.2 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 7. In order to correctly handle those cases we must take some care close

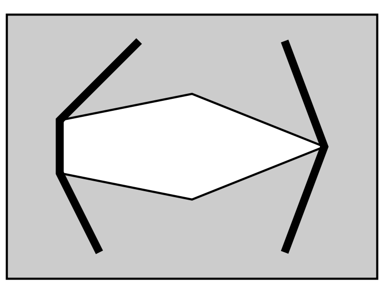


Fig. 7. Geodesics touching the boundary.

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_n^i 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_n^i.

3.3.3 Speeding up convergence.

In order to improve the performance of our algorithm, we explore a different correction strategy. We put all the interior vertices of the curve Γ_0 in a heap sorted by the error, see section 3.3.1. In each step the vertex 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.

Table 1
Run time and number of vertex corrections for the originally proposed method and the alternative method using a heap.

	Num. of	Original Method		Using a Heap	
	Vertices	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 1) the number of vertex corrections and the run time (in seconds) using each method. The program stopped when the error was smaller than 0.005. The number of vertex 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.

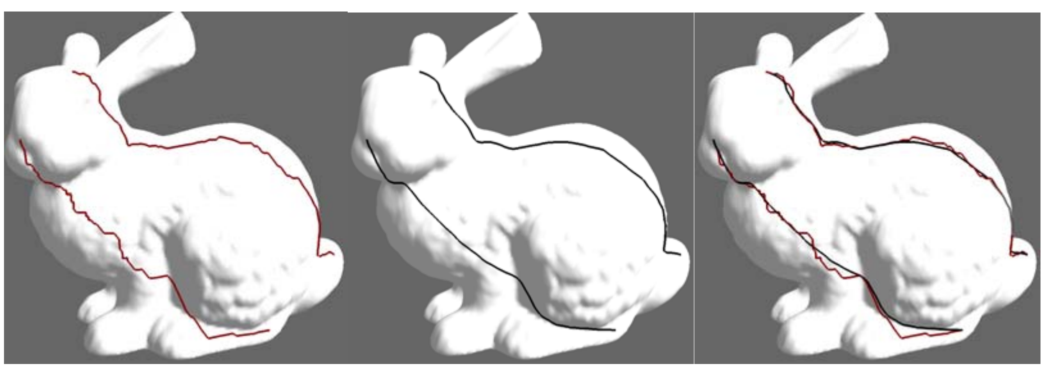
Consider the sequence $L(\Gamma_i)$ of curve lengths. A lower bound on the set $\{L(\Gamma_i),\ i = 0, 1, 2, \cdots\}$ is given by 0 since the length of Γ_i must be always positive. On the other hand, the length of Γ_i is reduced at every vertex correction, so we have the inequalities

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

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 conjecture 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. So far, however, we have only been able to prove this for the case where the surface is plane (see [13]). 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 figures in next section), and we can expect our final curve to be very close to a global minimizer of the length functional.

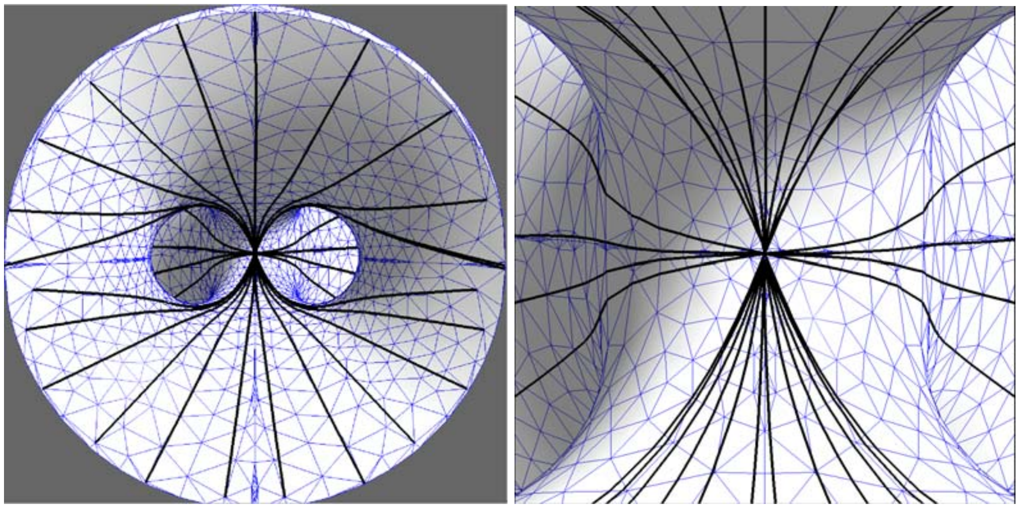
4 Experiments

In this section we show some results of our algorithm. In figures 8, 9 and 10 we show some geodesics over the Stanford bunny, Costa's surface and a face model. In figure 8 we also show the first approximations Γ_0 for the geodesics in the bunny model.



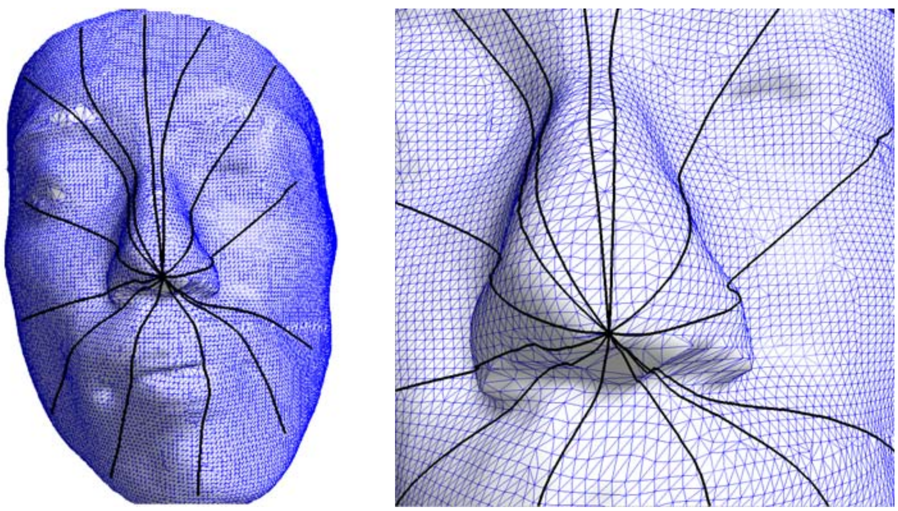
(a) (b) (c)

Fig. 8. Geodesics over the Stanford bunny. The first approximations (a), the computed geodesics (b), and both (c).



(a) (b)

Fig. 9. Some geodesics over Costa's Surface, all sharing a common extreme (a), and a zoom close to the common extreme (b).



(a) (b)

Fig. 10. Some geodesics over a Face model, all sharing a common extreme (a), and a zoom close to the common extreme (b).

4.1 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 1. After that, step 2 in algorithm 1 must be performed for every vertex of S.

Figure 11 shows some examples of the application of this algorithm to a sphere and to 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.

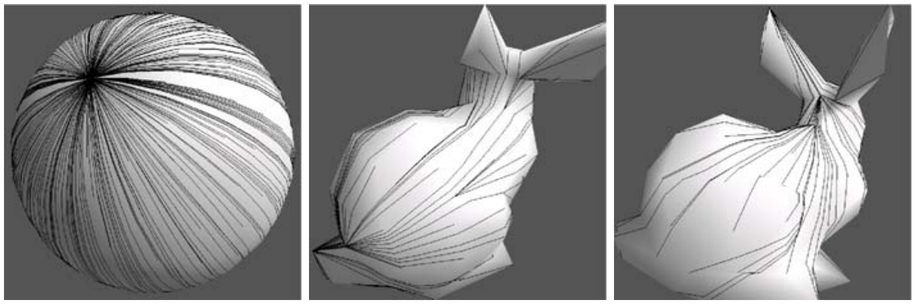


Fig. 11. Using our algorithm to solve single source problem.

5 Conclusions

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 authors know, there is not other algorithm which improves discrete geodesic approximations. It is 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.

Future work will study the convergence of the sequence of curves Γ_i as well as a generalization to non-manifold triangulations. We think it is also interesting to study other modifications in curve correction strategy in order to speed up the second step of our algorithm.

Acknowledgements

This research has been developed in the VISGRAF Laboratory at IMPA. VISGRAF Laboratory is sponsored by CNPq, FAPERJ, FINEP and IBM Brazil.

References

[1] M. S. Floater, K. Hormann, Parameterization of triangulations and unorganized points, in: A. Iske, E. Quak, M. S. Floater (Eds.), Tutorials on Multiresolution in Geometric Modelling, Mathematics and Visualization, Springer, 2002, pp. 287–316.

[2] O. Sifri, A. Sheffer, C. Gotsman, Geodesic-based surface remeshing, in: Proceedings of 12th International Meshing Roundtable, 2003.

[3] K. Polthier, M. Schmies, Geodesic flow on polyhedral surfaces, in: Data Visualization, Proceedings of Eurographics Workshop on Scientific Visualization, Springer Verlag, Vienna, 1999.

[4] A. D. Aleksandrov, V. A. Zalgaller, Intrinsic Geometry of Surfaces, Vol. 15 of Translation of Mathematical Monographs, AMS, 1967.

[5] K. Polthier, M. Schmies, Straightest geodesics on polyhedral surfaces, in: H.-C. Hege, K. Polthier (Eds.), Visualization and Mathematics, Springer Verlag, Heidelberg, 1998, pp. 135–150.

[6] J. S. B. Mitchell, D. M. Mount, C. H. Papadimitriou, The discrete geodesic problem, SIAM J. COMPUT. 16 (1987) 647–668.

[7] S. Kapoor, Efficient computation of geodesic shortest paths, in: Proceedings of 31st Annu. ACM Sympos. Theory Comput., 1999, pp. 770–779.

[8] J. Sethian, R. Kimmel, Computing geodesic paths on manifolds, Proceedings of the National Academy of Sciences of the USA 95 (15) 8431–8435.

[9] J. Chen, Y. Han, Shortest paths on a polyhedron, in: Proceedings of 6th Annu. ACM Sympos. Comput. Geom, 1990, pp. 360–369.

[10] V. Surazhsky, T. Surazhsky, D. Kirsanov, S. Gortler, H. Hoppex, Fast exact and approximate geodesics on meshes, submitted to ACM SIGGRAPH 2005.

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

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

[13] D. Martinez, P. C. Carvalho, L. Velho, Proof of convergence in the plane, unpublished Notes.
URL http://www.impa.br/∼dimasmm/geodesicas/proof.ps