Efficiently Computing Exact Geodesic Loops within Finite Steps

Shi-Qing Xin, Ying He, *Member*, *IEEE*, and Chi-Wing Fu, *Member*, *IEEE*

Abstract—Closed geodesics, or geodesic loops, are crucial to the study of differential topology and differential geometry. Although the existence and properties of closed geodesics on smooth surfaces have been widely studied in mathematics community, relatively little progress has been made on how to compute them on polygonal surfaces. Most existing algorithms simply consider the mesh as a graph and so the resultant loops are restricted only on mesh edges, which are far from the actual geodesics. This paper is the first to prove the existence and uniqueness of geodesic loop restricted on a closed face sequence; it contributes also with an efficient algorithm to iteratively evolve an initial closed path on a given mesh into an *exact* geodesic loop within finite steps. Our proposed algorithm takes only an $O(k)$ space complexity and an $O(mk)$ time complexity (experimentally), where m is the number of vertices in the region bounded by the initial loop and the resultant geodesic loop, and k is the average number of edges in the edge sequences that the evolving loop passes through. In contrast to the existing geodesic curvature flow methods which compute an approximate geodesic loop within a predefined threshold, our method is exact and can apply directly to triangular meshes without needing to solve any differential equation with a numerical solver; it can run at interactive speed, e.g., in the order of milliseconds, for a mesh with around $50K$ vertices, and hence, significantly outperforms existing algorithms. Actually, our algorithm could run at interactive speed even for larger meshes. Besides the complexity of the input mesh, the geometric shape could also affect the number of evolving steps, i.e., the performance. We motivate our algorithm with an interactive shape segmentation example shown later in the paper.

Index Terms—Discrete geodesic, geodesic loop, triangular mesh.

1 Introduction

Computing discrete geodesics on polygonal meshes plays an important role in many applications in computer graphics and computational geometry, such as surface parameterization [1], remeshing [2], nonrigid registration [3], shape segmentation [4], [5], [6], etc. However, a large body of past research literature [7], [8], [9] focused mainly on the geodesic path problem: computing the shortest path between two given points on a mesh. In this paper, we discuss how to compute closed geodesics, or *geodesic loops*. It is a very different problem compared with the above geodesic path problem: geodesic loops have no prescribed fixed points. Instead, we specify an initial closed curve as input to the problem and compute a new closed curve that is locally shortest and stable to disturbance, see Fig. 1.

In fact, closed geodesics on a smooth manifold have been widely studied in mathematics community [10], [11]. In recent years, researchers found that many computer graphics problems heavily depend on the computation of geodesic loops, for example, surface classification [12], shape segmentation [13], [14], and topological analysis (cutting an arbitrary genus surface into a single topological disk) [15]. However, very few progress has been made on this discrete geodesic loop problem. Typically, most existing algorithms [16], [17], [18] take the input mesh as a graph and therefore the resultant loops are restricted only on mesh edges, which are far from the actual geodesics. To our best knowledge, no existing algorithms can compute an exact geodesic loop so far.

The best known algorithm is based on the geodesic curvature flow methods. Wu and Tai [19] presented an algorithm to compute geodesic loops on polygonal meshes using discrete geodesic curvature flow (dGCF). By converting polygonal mesh to a level set, they formulated the geodesic curvature flow equation based on an energy minimization of curves, and then solved the equation by finite volume method. dGCF is very flexible and effective to compute geodesic loops on weighted graphs, thus, is promising for applications such as shape segmentation and feature detection. However, dGCF is computationally highly expensive because it involves solving a very large nonsymmetric linear system with the number of vertices as the problem dimension for each iteration. Later on, Zhang et al. [6] improved dGCF by dramatically reducing the problem dimension and proposed fast geodesic curvature flow (FGCF). This improved method can compute geodesic loops on meshes with $50K$ vertices in just a few seconds. However, both dGCF and FGCF must convert discrete meshes into a continuous representation and apply a numerical solver to solve a very large linear system. Furthermore, the geodesic curvature flow approach is an iterative algorithm that converges to the exact solution only when the evolution time goes to ∞. Thus, it usually has an early stop when the error tolerance is less than a user-specified threshold. In other words, the geodesic curvature flow can compute only *approximate* geodesic loops.

In this paper, we present an efficient algorithm to compute exact geodesic loops on triangular meshes. Taking a closed curve embedded on a given mesh as an initial loop, our algorithm gradually evolves it into an exact geodesic loop in finite number of iterations. It takes $O(mk)$ time

• *The authors are with the School of Computer Engineering, Nanyang Technological University, Singapore.*
E-mail: {sqxin, yhe, cwfu}@ntu.edu.sg.

Manuscript received 10 Dec. 2010; revised 7 Apr. 2011; accepted 8 Apr. 2011; published online 16 June 2011.
Recommended for acceptance by J. Stam.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2010-12-0290.
Digital Object Identifier no. 10.1109/TVCG.2011.119.

1077-2626/12/$31.00 © 2012 IEEE Published by the IEEE Computer Society

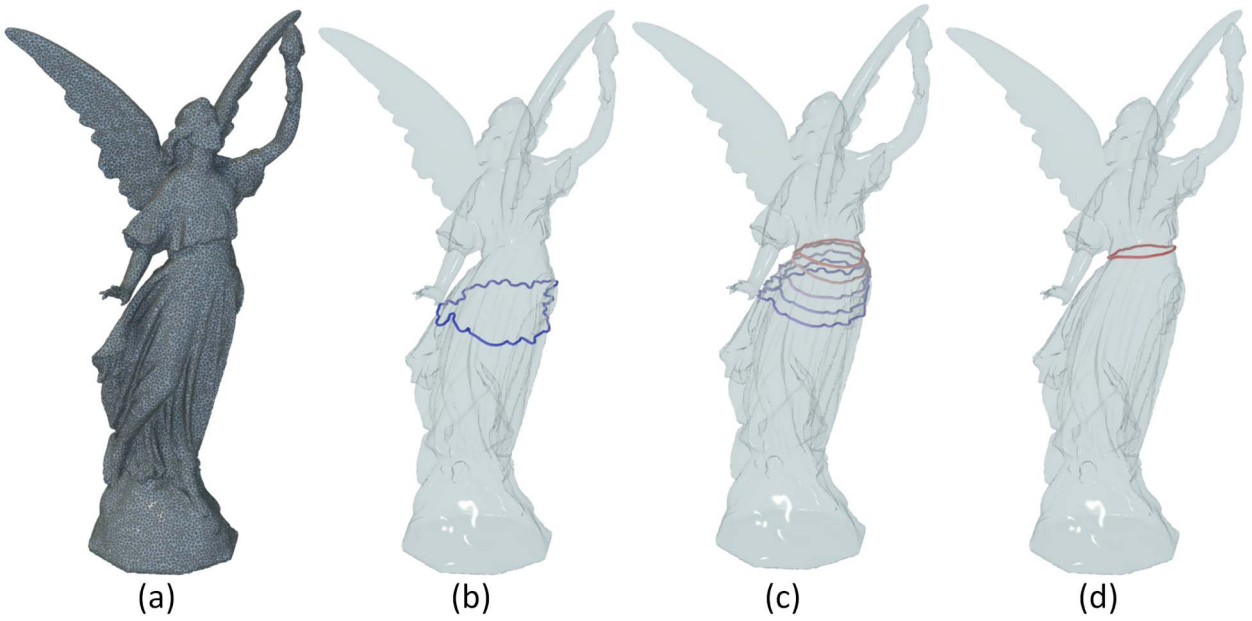


Fig. 1. Our algorithm can compute an exact geodesic loop in 15 milliseconds for this example. (a) An input mesh with $50K$ vertices. (b) A loop arbitrarily specified by the user as the initial loop. (c) A sequence of evolving loops (starting from the initial loop) with gradually reducing length. (d) Our proposed algorithm can generate the resultant geodesic loop in finite steps with the best performance.

(experimentally) and $O(k)$ space, where m is the number of vertices in the region bounded by the initial loop and resultant geodesic loop, and k is the average number of edges in the edge sequence that the evolving loop passes through. As our algorithm can apply directly to triangular meshes and does not require any numerical solver, it can run extremely fast in practice, e.g., in the order of milliseconds for meshes with around $50K$ vertices. Actually, our algorithm could run at interactive speed even for larger meshes. Besides the size of the input mesh, the geometric shape could also affect the number of evolving steps, i.e., the performance. Unlike existing geodesic curvature flow methods mentioned above, our algorithm is guaranteed to produce *exact* geodesic loops within finite iterations. Furthermore, it is intrinsic to geometry and robust to the mesh resolution and triangulation.

The remaining of the paper is organized as follows: Section 2 first reviews related work on discrete open and closed geodesics. After that, Section 3 presents some related terminologies for the discrete geodesic loop problem while Section 4 presents our algorithm to compute an exact geodesic loop on a triangular mesh. Section 5 shows our experimental results. Finally, Section 6 discusses the difference between our work and previous related work and Section 7 draws the conclusion.

2 Related Work

The discrete geodesic problem, i.e., the shortest path problem between two points on polyhedral surfaces, was first introduced by Sharir and Schorr [7], where an $O(n^3 \log n)$ algorithm was presented for convex polyhedra. Later, Mitchell et al. [8] (MMP) improved the time complexity bound to $O(n^2 \log n)$ by using the "continuous Dijkstra" technique, while Chen and Han [20] (CH) suggested building a binary tree to encode all the edge sequences that can possibly compute a shortest path, thereby improving the time complexity to $O(n^2)$. Surazhsky et al. [21] implemented the MMP algorithm and extended it to compute approximate geodesics with bounded error. Very recently, Xin and Wang [9] improved the CH algorithm by exploiting a filtering theorem; their proposed method outperforms both the MMP and CH algorithms.

Polthier and Schmies [22] introduced discrete geodesic curvature on polyhedral surfaces and defined discrete straightest geodesics that allow a unique solution of the initial value problem for geodesics. Later, they developed a method to compute the evolution of distance circles on polyhedral surfaces using geodesic flow [23]. Xin and Wang [24] presented an algorithm to compute a locally exact geodesic between two distinct vertices on a polyhedral surface. Note that Xin and Wang's algorithm does not generate a geodesic loop if we simply glue the starting and ending points, because the locally shortest condition does not hold at this point. In addition, many algorithms [25], [26], [27], [28] have been proposed to compute approximate discrete geodesics. Among these approximation algorithms, the Fast Marching Method [25] with an $O(n \log n)$ time complexity has been widely used in the research community.

However, all the above algorithms, whether exact or approximate, are mainly for computing the shortest path between two given points on a mesh. By contrast, the geodesic loop problem is very different and more complicated because we do not have prescribed fixed points during the computational process. The resultant geodesic loop could be quite far away from the initial input loop; see Fig. 1. Therefore, it is impractical to directly adapt the above algorithms to the geodesic loop problem.

To our knowledge, most existing algorithms [16], [17], [29] are graph-based and therefore the resultant loops are restricted only on mesh edges, which are far from the actual geodesics. Very few algorithms are known for computing geodesic loops. Wu and Tai [19] proposed the discretized geodesic curvature flow (dGCF) to compute geodesic loops on triangular meshes using a level set

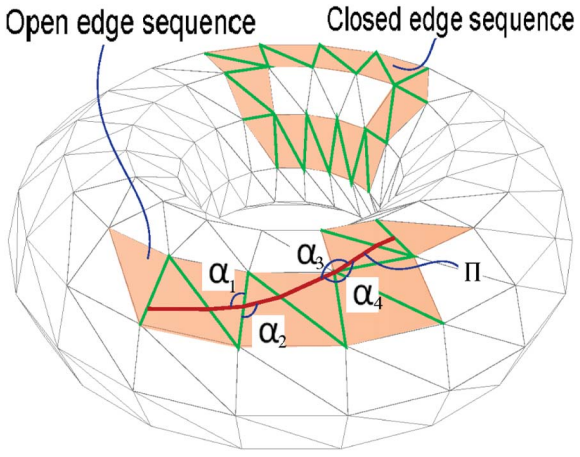


Fig. 2. Edge sequence, face sequence, and geodesic paths.

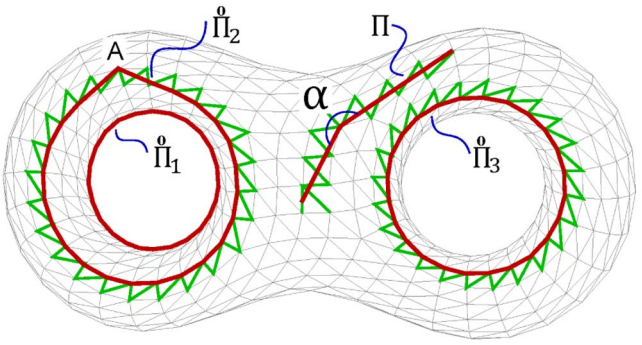


Fig. 3. Examples of open and closed geodesic paths. Π: an open geodesic path; $\mathring{\Pi}_1$: a geodesic loop on the mesh; $\mathring{\Pi}_2$: a relaxed geodesic loop (at vertex A) restricted on an edge sequence; and $\mathring{\Pi}_3$: a geodesic loop restricted on an edge sequence. Geodesics are drawn in red and the corresponding edge sequences are drawn in green.

formulation, and later, Zhang et al. [6] improved dGCF to FGCF by reducing the problem dimension.

As mentioned in the introduction, very few algorithms, e.g., dGCF [19] and FGCF [6], are known for the geodesic loop problem. Other than using geodesic curvature flow, another approach is by Zeng et al. [30], where they presented a technique to compute the shortest loop for each homotopy class of oriented closed surfaces of high genus. Their method requires computing the uniformization metric and then embeds the surface into a universal covering space. The uniformization metric is computed by discrete Ricci flow or Yamabe flow [31]. Since both curvature flows are highly nonlinear PDEs, the computational costs are very high. Compared to this approach, our method is computationally efficient and numerically stable, and it can work for large meshes, see Section 5. Furthermore, the computed geodesics using [30] are in terms of the uniformization metric rather than the original euclidean metric.

Another topic related to our work is the transformation of curves on polygonal surfaces. Dey and Guha [16] proposed methods to determine if two closed curves are homotopic. Diaz-Gutierrez et al. [29] presented a graph-based algorithm to find fundamental cycles aligned with the principal curvature directions on a surface. Though these techniques deal with curves on surfaces, they are graph-based, and the resultant loops are restricted only on mesh edges; hence, their results are far from the actual geodesics.

There is also a large body of literatures in surface segmentation by using closed curves. Some recent work includes random walks [32], mesh snapping [6], cross-boundary brushes [33], randomized cuts [34], and paint mesh cutting [18]. These approaches usually compute the "shortest" loop on a weighted graph by optimization or geodesic curvature flow, while our method computes the geodesic loop based on the original metric without needing any numerical solver. As discussed in Section 5, our method can also be used to support interactive shape segmentation.

3 Preliminaries

In this section, we introduce a set of definitions and review some basic properties on discrete geodesics from [7], [8], and [20].

Definition 1. *A* face sequence $\mathcal{F}$ *is an ordered list of faces, say* $f_1, f_2, \ldots, f_{k+1}$, *such that each pair of consecutive faces, i.e.,* f_i *and* f_{i+1}, *shares a common edge, say* e_i. *The associated list of shared edges, say* $\mathcal{E} = (e_1, e_2, \ldots, e_k)$, *is called an* edge sequence.

Definition 2. *An edge sequence is* simple *if it contains no repeated edges except for the first edge* e_1 *and the last edge* e_k, *which may be the same.* $\mathcal{E}$ *is* open *if* $e_1 \neq e_k$, closed *otherwise.*

Throughout this paper, the open and closed edge sequences are denoted by $\mathcal{E}$ and $\mathring{\mathcal{E}}$, respectively, see Fig. 2 for example edge sequences. Unless specified, all edge sequences mentioned in the rest of this paper are simple. In addition, the terms *vertex* and *edge* in this paper refer to vertex and edge on the given mesh.

Planar unfolding is a widely used technique for computing shortest paths on polygonal meshes. A face sequence $\mathcal{F}$ can be unfolded in this way: rotate f_2 around e_1 to make f_1 and f_2 coplanar; rotate f_3 around e_2 to make f_2 and f_3 coplanar; and repeat this until all faces are coplanar. Obviously, this process takes $O(k)$ time, where k is the number of faces in $\mathcal{F}$.

In the continuous setting, geodesics have three equivalent definitions: locally shortest, locally straightest, and the curvature vector being parallel to the surface normal. These definitions are, however, not equivalent in the discrete setting. Among them, *locally straightest* geodesics [22] and *locally shortest* geodesics [7] have been widely studied. In this paper, we adopt the popular choice of defining a discrete geodesic as a polyline path that is *locally shortest everywhere* on a given mesh M; see Π in Fig. 2 and $\mathring{\Pi}_1$ in Fig. 3. In this formulation, a path Π (whether open or closed) on M is said to be a *geodesic* if and only if the following conditions are all satisfied (from [7]):

- For each face (on M) that Π intersects, the intersection must be a single point or a straight line segment on the face.
- For each edge (on M) that Π intersects, the entry and departure angles must be the same; as shown in Fig. 2, $\alpha_1 = \alpha_2$.
- For each vertex (on M) that Π passes through, the angles on the two sides of Π at the vertex must be greater than or equal to π; as shown again in Fig. 2, $\alpha_3 \geq \pi$ and $\alpha_4 \geq \pi$.

For the general form of a locally shortest geodesic path, we refer the readers to [8].

Lemma 1 [8]. *Given two points, say A and B, on a mesh M, the general form of a locally shortest path $\Pi(A,B)$ between A and B is an alternating sequence of vertices and edge sequences on M, such that the unfolded path for each edge sequence is a straight line segment joining corresponding vertices; in addition, the angles on the two sides of $\Pi(A,B)$ at each vertex that $\Pi(A,B)$ passes through must be greater than or equal to π (locally shortest). In this way, $\Pi(A,B)$ is said to be a* geodesic path *between A and B; see Π in Fig. 2 as an example, which has two edge sequences connected by one vertex on M.*

Definition 3. *An open (resp. closed) path Π (resp. $\mathring{\Pi}$) is said to be a* restricted *geodesic on an edge sequence $\mathcal{E}$ (resp. $\mathring{\mathcal{E}}$), or an $\mathcal{E}$-*restricted *(resp. $\mathring{\mathcal{E}}$-*restricted*) geodesic path, if and only if the following conditions are all satisfied:*

- *Π (resp. $\mathring{\Pi}$) is inside the face sequence associated with $\mathcal{E}$ (resp. $\mathring{\mathcal{E}}$).*
- *Π (resp. $\mathring{\Pi}$) is locally shortest everywhere, except at vertices that Π (resp. $\mathring{\Pi}$) passes through, where the angle (at each of these vertices) facing $\mathcal{E}$ (resp. $\mathring{\mathcal{E}}$), e.g., α in Fig. 3, is always greater than or equal to π (note that we only need to consider angles facing $\mathcal{E}$ but not those on the other side). We also say that Π (resp. $\mathring{\Pi}$) is* inward convex *at all the vertices it passes through.*

Definition 4. *A closed path $\mathring{\Pi}$ that passes through a vertex A on M is said to be an A-*relaxed *geodesic loop if and only if it is locally shortest everywhere except at A. See $\mathring{\Pi}_2$ in Fig. 3 as an example.*

The following show the notations used in the rest of the paper:

- $\mathcal{E}, \mathring{\mathcal{E}}$: an open/closed edge sequence;
- $\mathcal{F}, \mathring{\mathcal{F}}$: an open/closed face sequence;
- Π, $\Pi(A,B)$: an open path (resp. between points A and B);
- $\mathring{\Pi}$: a geodesic loop;
- $\mathring{\Pi}_{\mathring{\mathcal{E}}}$: a geodesic loop restricted on a closed edge sequence $\mathring{\mathcal{E}}$;
- $\mathring{\Pi}^{A}$: an A-relaxed geodesic loop;
- $\mathring{\Pi}^{A}_{\mathring{\mathcal{E}}}$: an $\mathring{\mathcal{E}}$-restricted and A-relaxed geodesic loop; and
- $\|\Pi\|, \|\mathring{\Pi}\|$: length of the path/loop.

4 Geodesic Loop Algorithm

This section presents the algorithm to evolve an initial closed curve to an exact geodesic loop (or sometimes a degenerate point). It proceeds by alternatively performing two operations: compute the shortest loop restricted on a given closed edge sequence and update the edge sequence so that it can further lead to a shorter loop.

4.1 Computing the Shortest Loop Restricted on $\mathring{\mathcal{E}}$

Lemma 2. *Given an open edge sequence $\mathcal{E} = (e_1, e_2, \ldots, e_k)$ and two points A, B on its associated face sequence, the $\mathcal{E}$-restricted geodesic path from A to B, say $\Pi_{\mathcal{E}}(A,B)$, is always unique and therefore the shortest among all possible $\mathcal{E}$-restricted paths from A to B. Furthermore, $\Pi_{\mathcal{E}}(A,B)$ can be computed in $O(k)$ time.*

Proof. Upon unfolding, the face sequence $\mathcal{F}$ associated with $\mathcal{E}$ covers a polygon area $\mathcal{G}$ (which may overlap upon unfolding) with a single boundary. Therefore, computing the restricted geodesic path between A and B is equivalent to determine the shortest path from A to B inside the planar polygon $\mathcal{G}$. From [35], this has been proved, and the result still holds even if the unfolded image of $\mathcal{F}$ has overlap in the 2D domain. □

Lemma 3. *Given a closed edge sequence $\mathring{\mathcal{E}}$ (associated with the face sequence $\mathring{\mathcal{F}}$) and a vertex $A \in \mathring{\mathcal{F}}$, there exists a unique A-relaxed $\mathring{\mathcal{E}}$-restricted geodesic loop.*

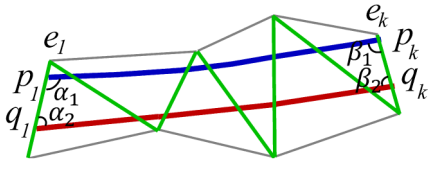
Proof. We can cut $\mathring{\mathcal{E}}$ at vertex A into an open edge sequence $\mathcal{E}$ with the two edges (in $\mathring{\mathcal{E}}$) connecting at A being the first and last edges of $\mathcal{E}$; see $\mathring{\Pi}_2$ in Fig. 3 as an example. By Lemma 2, there must be a unique $\mathcal{E}$-restricted geodesic path from point A_1 to A_2, where A_1 and A_2 are copies of A on different ends of $\mathcal{E}$. Since such a path is locally shortest everywhere on $\mathcal{E}$, if we sew $\mathcal{E}$ back to $\mathring{\mathcal{E}}$, the path becomes the unique A-relaxed $\mathring{\mathcal{E}}$-restricted geodesic loop, which is locally shortest everywhere on $\mathring{\mathcal{E}}$, except at A. □

Lemma 4. *Given a closed edge sequence $\mathring{\mathcal{E}}$, the $\mathring{\mathcal{E}}$-restricted geodesic loop (see Definition 3) is the shortest in length among all loops that pass through every single edge in $\mathring{\mathcal{E}}$. Furthermore, the $\mathring{\mathcal{E}}$-restricted geodesic loop is unique (up to a* translation *within the face sequence associated with $\mathring{\mathcal{E}}$).*

Proof. First, it is worth to note that we can prove the above lemma by showing that if there exist two different $\mathring{\mathcal{E}}$-restricted geodesic loops $\mathring{\Pi}^1_{\mathring{\mathcal{E}}}$ and $\mathring{\Pi}^2_{\mathring{\mathcal{E}}}$, then $\|\mathring{\Pi}^1_{\mathring{\mathcal{E}}}\| = \|\mathring{\Pi}^2_{\mathring{\mathcal{E}}}\|$.

And before we proceed, we first show that $\mathring{\Pi}^1_{\mathring{\mathcal{E}}}$ and $\mathring{\Pi}^2_{\mathring{\mathcal{E}}}$ cannot intersect: assume they have an intersection point, say s. We can cut[1] $\mathring{\mathcal{E}}$ at s into an open edge sequence $\mathcal{E}$. Then, we obtain two $\mathcal{E}$-restricted geodesic paths from s_1 to s_2, where s_1 and s_2 are copies of s on different ends of $\mathcal{E}$. This contradicts to the uniqueness of $\mathcal{E}$-restricted geodesic path between two fixed points, see Lemma 2.

In the following, we assume $\mathring{\Pi}^1_{\mathring{\mathcal{E}}}$ and $\mathring{\Pi}^2_{\mathring{\mathcal{E}}}$ intersect a certain edge $e_1 \in \mathring{\mathcal{E}}$ at p_1 and q_1, respectively. Upon cutting $\mathring{\mathcal{E}}$ at e_1 and planar unfolding $\mathring{\mathcal{E}}$, we have e_k, p_k, q_k as copies of e_1, p_1, q_1 on the other end of the unfolded image, see the figure below.



Without loss of generality, we assume that $\mathring{\Pi}^1_{\mathring{\mathcal{E}}}$ is closer to the upper boundary of the unfolded image than $\mathring{\Pi}^2_{\mathring{\mathcal{E}}}$, see the figure above. Now, we define the four angles, α_1, α_2, β_1, and β_2, as shown again in the figure. As proved previously that $\mathring{\Pi}^1_{\mathring{\mathcal{E}}}$ and $\mathring{\Pi}^2_{\mathring{\mathcal{E}}}$ cannot intersect, $\mathring{\Pi}^1_{\mathring{\mathcal{E}}}$ must not pass through any vertex on the lower boundary. Therefore, $\mathring{\Pi}^1_{\mathring{\mathcal{E}}}$ (resp. $\mathring{\Pi}^2_{\mathring{\mathcal{E}}}$) on the unfolded

1. If s is at a vertex in $\mathring{\mathcal{E}}$, we cut at the vertex so that the two connected edges in $\mathring{\mathcal{E}}$ are separated; if s is on an edge in $\mathring{\mathcal{E}}$, we cut this edge into two copies; and if s is inside a face, we separate the two edges (in $\mathring{\mathcal{E}}$) that are on this face.

image must be a straight line segment or a convex polyline bending upward (resp. downward) at the vertices that it passes through. Hence, we have[2]

$$\alpha_1 + \beta_1 + \alpha_2 + \beta_2 \leq 2\pi.$$

Since $\mathring{\Pi}^1_{\mathring{\mathcal{E}}}$ and $\mathring{\Pi}^2_{\mathring{\mathcal{E}}}$ are $\mathring{\mathcal{E}}$-restricted geodesic loops, the following inequalities hold (according to Definition 3):

$$\begin{cases} \alpha_1 + \beta_1 \geq \pi, \\ \alpha_2 + \beta_2 \geq \pi. \end{cases}$$

Putting the above inequalities together, we have

$$\begin{cases} \alpha_1 + \beta_1 = \pi, \\ \alpha_2 + \beta_2 = \pi. \end{cases}$$

Since the above inequalities hold for every edge in $\mathring{\mathcal{E}}$, the entry angle always equals to the departure angle for each edge that $\mathring{\Pi}^1_{\mathring{\mathcal{E}}}$ and $\mathring{\Pi}^2_{\mathring{\mathcal{E}}}$ pass through, and the area bounded by $e_1, e_k, \mathring{\Pi}^1_{\mathring{\mathcal{E}}}$, and $\mathring{\Pi}^2_{\mathring{\mathcal{E}}}$ must form a quadrilateral on the unfolded image. Furthermore, since $\alpha_1 + \beta_1 = \pi$ and $\|\overline{p_1 q_1}\| = \|\overline{p_k q_k}\|$, $p_1 p_k q_k q_1$ must be a parallelogram. This shows that $\|\mathring{\Pi}^1_{\mathring{\mathcal{E}}}\| = \|\mathring{\Pi}^2_{\mathring{\mathcal{E}}}\|$.

To summarize, an $\mathring{\mathcal{E}}$-restricted geodesic loop is unique (up to a translation within the closed face sequence associated with $\mathring{\mathcal{E}}$). □

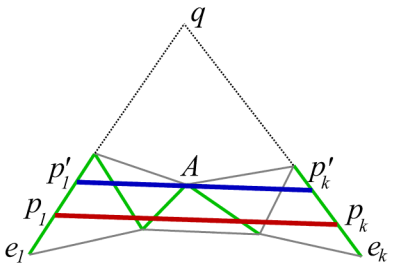
Corollary 5. *Given a closed edge sequence $\mathring{\mathcal{E}}$ and an A-relaxed geodesic loop $\mathring{\Pi}^{-A}_{\mathring{\mathcal{E}}}$, $\mathring{\Pi}^{-A}_{\mathring{\mathcal{E}}}$ is the shortest loop restricted on $\mathring{\mathcal{E}}$ if and only if $\alpha \geq \pi$, where α is the angle subtended by $\mathring{\Pi}^{-A}_{\mathring{\mathcal{E}}}$ at A and facing $\mathring{\mathcal{E}}$.*

Proof. $\Pi^{-A}_{\mathring{\mathcal{E}}}$ is the shortest loop $\Longleftrightarrow$ it is locally shortest everywhere as well as at A $\Longleftrightarrow \Pi^{-A}_{\mathring{\mathcal{E}}}$ is the A-relaxed geodesic loop on $\mathring{\mathcal{E}}$ and $\alpha \geq \pi$ (at A). □

Lemma 6. *Given a closed edge sequence $\mathring{\mathcal{E}}$, there always exists a relaxed geodesic loop $\mathring{\Pi}^{-A}_{\mathring{\mathcal{E}}}$ at some vertex A such that it is the shortest among all loops that pass through every single edge in $\mathring{\mathcal{E}}$.*

Proof. If the shortest loop $\mathring{\Pi}_{\mathring{\mathcal{E}}}$ on $\mathring{\mathcal{E}}$ passes through a certain vertex, then the conclusion naturally holds (from Corollary 5). In the following, we assume that $\mathring{\Pi}_{\mathring{\mathcal{E}}}$ does not pass through any vertex in $\mathring{\mathcal{E}}$. Let $e \in \mathring{\mathcal{E}}$ be an edge and p be the intersection point between $\mathring{\Pi}_{\mathring{\mathcal{E}}}$ and e. We can cut $\mathring{\mathcal{E}}$ at e and unfold $\mathring{\Pi}_{\mathring{\mathcal{E}}}$ into a straight line segment[3] as shown in the figure above, where e_1 and e_k are the unfolded images of e on different ends, and correspondingly, $p_1 \in e_1$ and $p_k \in e_k$ are unfolded images of p.

If e_1 and e_k are not parallel, the extended straight lines of e_1 and e_k must intersect at a certain point, say q. Observing that $\triangle p_1 p_k q$ encloses at least one vertex of $\mathring{\mathcal{E}}$ (for example, the upper endpoint of e is inside $\triangle p_1 p_k q$ in the above figure), we can move p_1 and p_k to p'_1 and p'_k toward q, respectively, by a certain distance such that the resultant path $\overline{p'_1 p'_k}$ just touches a certain nearest vertex, say A. From Lemma 8 in the Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2011.119, we know that $\overline{p'_1 p'_k}$ is always shorter than $\mathring{\Pi}_{\mathring{\mathcal{E}}}$. This contradicts to the assumption that $\mathring{\Pi}_{\mathring{\mathcal{E}}}$ is the shortest loop restricted on $\mathring{\mathcal{E}}$.



In fact, if e_1 and e_k are parallel, we can still move $\mathring{\Pi}_{\mathring{\mathcal{E}}}$ without increasing its length, so that it passes through a certain vertex. □

Theorem 7. *The exact shortest loop restricted on a given closed edge sequence $\mathring{\mathcal{E}}$ exists and is unique up to a translation within $\mathring{\mathcal{E}}$. It can be computed in $O(k)$ space and worst case $O(k^2)$ time, where k is the number of edges $\mathring{\mathcal{E}}$ contains.*

Proof. The existence and uniqueness are immediate results from Lemma 4, Corollary 5, and Lemma 6. As for the time complexity, we use Guibas and Hershberger's algorithm [35] to compute relaxed geodesic loops for all the vertices restricted on $\mathring{\mathcal{E}}$, in $O(k^2)$ time and $O(k)$ space. The $\mathring{\mathcal{E}}$-restricted shortest loop is the shortest loop among them. Therefore, the time and space complexities are $O(k^2)$ and $O(k)$, respectively. □

Theorem 7 implies an $O(k^2)$-time naïve algorithm to compute the $\mathring{\mathcal{E}}$-restricted shortest loop. In fact, there are some effective techniques to improve the complexity to $O(k)$ in practice. In the following, we list some speedup techniques:

- Let $v \in \mathring{\mathcal{E}}$ be a vertex, and θ_1 and θ_2 be two angles that the v-relaxed $\mathring{\mathcal{E}}$-restricted geodesic loop $\mathring{\Pi}^{-v}_{\mathring{\mathcal{E}}}$ incident at v. Without loss of generality, assume that θ_1 is the angle facing $\mathring{\mathcal{E}}$. Hence, if $\theta_1 \geq \pi$, $\mathring{\Pi}^{-v}_{\mathring{\mathcal{E}}}$ must be the shortest $\mathring{\mathcal{E}}$-restricted loop, see Corollary 5.
- Let e be the shortest edge in $\mathring{\mathcal{E}}$. The endpoints of e should be given higher priority when computing the relaxed shortest loops since the $\mathring{\mathcal{E}}$-restricted shortest loop is more likely to pass through one of these endpoints.
- Let v be a vertex in $\mathring{\mathcal{E}}$. If the v-relaxed geodesic loop $\mathring{\Pi}^{-v}_{\mathring{\mathcal{E}}}$ is not the shortest restricted loop and it goes through another vertex v' that is on the opposite boundary (upper/lower) of $\mathring{\mathcal{E}}$, then the v'-relaxed geodesic loop $\mathring{\Pi}^{-v'}_{\mathring{\mathcal{E}}}$ must be the $\mathring{\mathcal{E}}$-restricted shortest loop we want; see Lemma 10 in the Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2011.119.

Summarizing the above ideas, we propose Algorithm 1 to compute the shortest restricted loop on a given closed edge sequence.

2. We can connect p_1, p_k, q_k, q_1 into a quadrilateral. It's known that the sum of the interior angles equals to 2π. Since $\alpha_1, \alpha_2, \beta_1, \beta_2$ are less than or equal to the corresponding interior angles, $\alpha_1 + \beta_1 + \alpha_2 + \beta_2 \leq 2\pi$ holds. More rigorous proof depends on the classical theorem of Kakutani [36].

3. If a geodesic path does not pass through any vertex, we can always unfold it into a straight line segment, see Lemma 1.

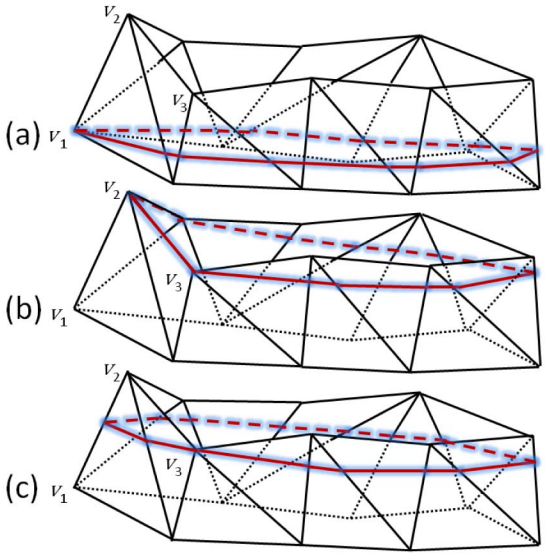


Fig. 4. Example: Running **Algorithm 1** on a face sequence, note the dashed lines for hidden faces. (a) We compute v_1-relaxed geodesic loop $\mathring{\Pi}_{\mathring{\mathcal{E}}}^{-v_1}$, where $v_1 \in e$ is an arbitrary vertex. However, $\mathring{\Pi}_{\mathring{\mathcal{E}}}^{-v_1}$ only passes through v_1 but not other vertices, where the path is not inward convex. (b) We switch to the other endpoint v_2 of edge e and compute v_2-relaxed geodesic loop $\mathring{\Pi}_{\mathring{\mathcal{E}}}^{-v_2}$. Since $\mathring{\Pi}_{\mathring{\mathcal{E}}}^{-v_2}$ is not inward convex at v_2 and passes through another vertex, which is v_3, we switch from v_2 to v_3. (c) Then, we can compute v_3-relaxed geodesic loop $\mathring{\Pi}_{\mathring{\mathcal{E}}}^{-v_3}$. Since the path is inward convex at v_3, $\mathring{\Pi}_{\mathring{\mathcal{E}}}^{-v_3}$ must be the shortest loop restricted on $\mathring{\mathcal{E}}$.

Algorithm 1.

Initialization: Suppose that e_0 is the shortest edge in $\mathring{\mathcal{E}}$ and A_0 is an endpoint of e_0. Let $e := e_0$ and $v := A_0$.

Step 1: Use Guibas and Hershberger's algorithm [35] to compute the v-relaxed geodesic loop $\mathring{\Pi}_{\mathring{\mathcal{E}}}^{-v}$ on $\mathring{\mathcal{E}}$ by cutting $\mathring{\mathcal{E}}$ at e and applying Lemma 2.

Step 2: Evolve $\mathring{\Pi}_{\mathring{\mathcal{E}}}^{-v}$ by the following substeps:

Step 2.1 Given α as the incident angle that $\mathring{\Pi}_{\mathring{\mathcal{E}}}^{-v}$ passes through at v and α faces $\mathring{\mathcal{E}}$. If $\alpha \geq \pi$, then output $\mathring{\Pi}_{\mathring{\mathcal{E}}}^{-v}$, see Corollary 5;

Step 2.2 If $\mathring{\Pi}_{\mathring{\mathcal{E}}}^{-v}$ passes through a certain vertex, say v', on the opposite boundary of $\mathring{\mathcal{E}}$, compute and output the v'-relaxed geodesic loop, see Lemma 10;

Step 2.3 If $\mathring{\Pi}_{\mathring{\mathcal{E}}}^{-v}$ does not pass through any other vertex, update v to be the other endpoint of e and goto Step 1, see Lemma 11[4];

Step 2.4 Otherwise, let v'' be another vertex $\mathring{\Pi}_{\mathring{\mathcal{E}}}^{-v}$ passes through. Update v to v'' (v'' and v are on the same boundary of $\mathring{\mathcal{E}}$ but v'' can lead to a shorter relaxed geodesic loop than v) and goto Step 1, see Lemma 9.

Fig. 4 illustrates Algorithm 1 with an example. The algorithm is much faster than the naïve version since the candidate vertices involved are only a subset of vertices in $\mathring{\mathcal{E}}$. Although the theoretical worst case upper bound of Algorithm 1 is still $O(k^2)$ (see Theorem 7), the average-case time complexity could be reduced to $O(k)$. Our experimental results show that when computing the restricted shortest loop on a given edge sequence, the average number of times to run Guibas and Hershberger's algorithm [35] is no more than 2. In other words, the time complexity is $O(k)$ experimentally, where k is the average number of edges in $\mathring{\mathcal{E}}$.

4. An alternative way is to find the "nearest" vertex, say C, in Lemma 11; then compute and output the C-relaxed geodesic loop as the result. In fact, this step works only when the evolving loop and the final restricted shortest loop pass through different boundaries of $\mathring{\mathcal{E}}$. In this case, updating the candidate vertex from one endpoint to the other moves the evolving loop to the proper boundary. Hence, Step 2.3 is performed at most once.

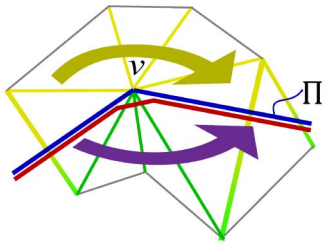


Fig. 5. If one of the angles at which path Π (the blue polyline) passes through vertex v is less than π, Π can always be shortened by perturbation at v, see the red polyline. Hence, the green edge sequence can lead to a shorter path than the yellow edge sequence.

4.2 Updating the Closed Edge Sequence

Given an initial closed edge sequence $\mathring{\mathcal{E}}_0$, we can compute the restricted shortest loop $\mathring{\Pi}_{\mathring{\mathcal{E}}_0}$ using Algorithm 1. In this section, we show how to update the edge sequence to obtain another closed edge sequence that can lead to a shorter loop.

To check if the resultant restricted shortest loop is locally shortest (stable to disturbance) on mesh M, it is sufficient to check if the path is locally shortest at every vertex that $\mathring{\Pi}_{\mathring{\mathcal{E}}_0}$ passes through. If both angles are greater than or equal to π at every vertex that $\mathring{\Pi}_{\mathring{\mathcal{E}}_0}$ passes through, $\mathring{\Pi}_{\mathring{\mathcal{E}}_0}$ is locally shortest on M (by Definition 2) and the curve evolution can stop. Otherwise, we can shorten $\mathring{\Pi}_{\mathring{\mathcal{E}}_0}$ by disturbing it at v, as shown in Fig. 5. Hence, we need to update $\mathring{\mathcal{E}}_0$ to a new edge sequence $\mathring{\mathcal{E}}_1$ that contains the green edges. Then, we can compute the shortest loop $\mathring{\Pi}_{\mathring{\mathcal{E}}_1}$ on $\mathring{\mathcal{E}}_1$ and obtain a shorter loop. Thus, we get a series of evolving loops $\{\mathring{\Pi}_{\mathring{\mathcal{E}}_0}, \mathring{\Pi}_{\mathring{\mathcal{E}}_1}, \mathring{\Pi}_{\mathring{\mathcal{E}}_2}, \ldots\}$, where $\mathring{\Pi}_{\mathring{\mathcal{E}}_{i+1}}$ is strictly shorter than $\mathring{\Pi}_{\mathring{\mathcal{E}}_i} (i \geq 0)$.

Since the number of edge sequences on a given mesh is finite and each edge sequence update always produces a shorter loop, we conclude that our algorithm can converge to an exact geodesic loop (or possibly a degenerate point) after a finite number of iterations. Note that during each iteration, we can simultaneously update the edge sequence locally at *all* vertices with which the path is not locally shortest. Fig. 6 shows an example of iteratively evolving an initial loop to an exact geodesic loop (which is a degenerate point).

5 Experimental Results

This section presents a series of extensive experiments to demonstrate the algorithm's efficiency, as well as its insensitiveness to mesh resolution and triangulation.

5.1 Performance

To demonstrate the algorithm's efficiency, we develop a simple interface for users to interactively specify an arbitrary cutting plane (hence, an initial loop) on a given triangular mesh. Our system can automatically evolve the given loop into an exact geodesic loop (or a degenerate point), see Fig. 7. To measure the algorithm performance, we randomly generate 1,000 cutting planes for each model, resulting in 1,000 or more initial loops. Then, we compute the average running time for our algorithm to evolve these loops. As shown in Table 1, for meshes with around $50K$

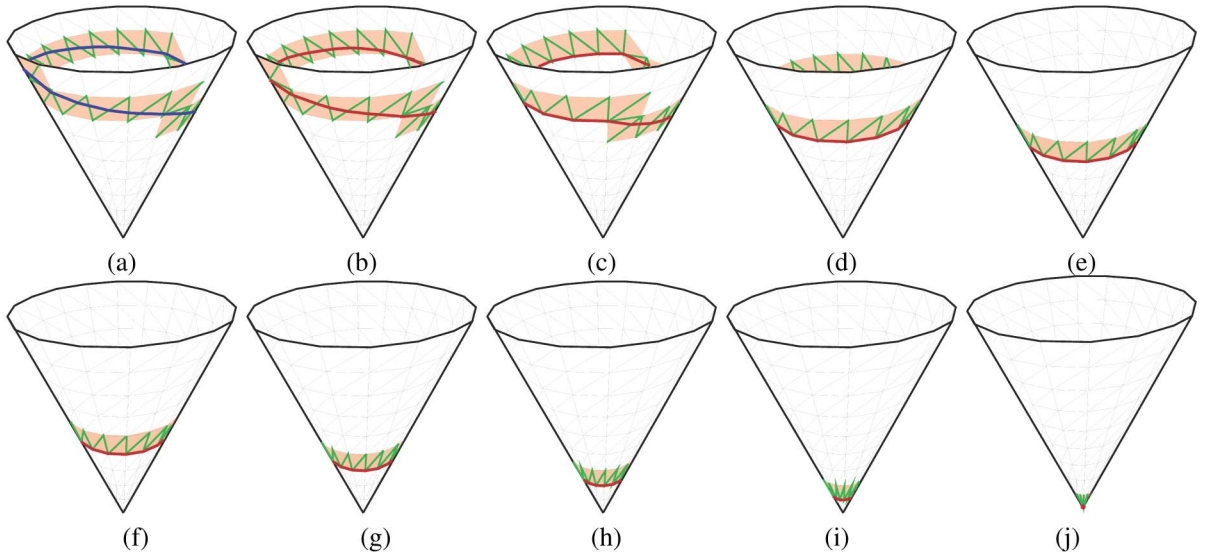


Fig. 6. Starting from a given closed curve, our method evolves the initial loop into an exact geodesic loop or a degenerate point in finite iterations. (a) A discretized cone of 306 triangles and an initial loop (in blue). (b) to (i) show the intermediate results after one to eight iterations and (j) shows the final degenerate point after nine iterations. Edge sequences are drawn in green and evolving loops in red.

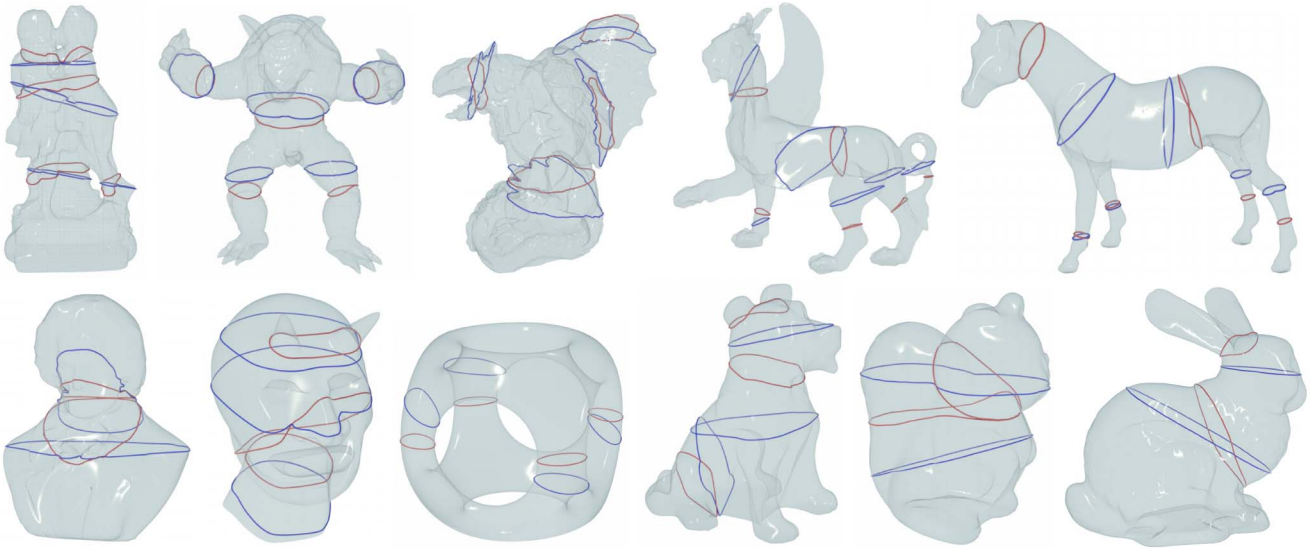


Fig. 7. From left to right and then top to bottom: Two Kids, 49,996 vertices; Armadillo, 172,974 vertices; Gargoyle, 349,998 vertices; Feline, 404,998 vertices; Horse, 48,484 vertices; Beethoven, 10,406 vertices; Oni, 22,807 vertices; Decocube, 19,992 vertices; Dog, 75,002 vertices; Squirrel, 13,280 vertices; Bunny, 72,027 vertices. The initial loops and resultant geodesic loops are drawn in blue and red, respectively.

vertices, our current unoptimized implementation can run at interactive speed, i.e., in the order of milliseconds. For each run in the experiment, the position of the initial loop and the geometric structure of the given mesh are also important factors affecting the computational cost.

Fig. 10 demonstrates interactive segmentation on Bunny using our proposed geodesic loop algorithm.

5.2 Time Complexity Analysis

To analyze the time complexity, we use the Dragon model with four different resolutions: $5K$, $10K$, $50K$, and $750K$ vertices, but similar mesh quality, see Fig. 8. We randomly generate 1,800 cutting planes and run our algorithm on the four models with the same cutting planes. The most important statistical indicators are m and k, where

- m: the number of vertices in the region bounded by the initial loop and the resultant geodesic loop; and
- k: the average number of edges in the edge sequences that the evolving loop passes through.

From the experimental results in Fig. 8, we observe that

1. The average number of times to run Guibas and Hershberger's algorithm [35] is no more than 2.
2. The number of evolving steps required is roughly linear to m.
3. The running time linearly depends on mk.

As Table 1 shows, the running time roughly increases linearly with the number of vertices in the Dragon models (though this rate appears to drop a little bit for the largest resolution model). We suggest two possible reasons for this. First, the evolving loop can move more steadily on finer meshes while there could be more "oscillations" for the case of coarser meshes. Second, there are more locally shortest closed curves on high-resolution meshes, especially when it has many detailed features.

TABLE 1
Timing Statistics (Seconds) Measured on a Workstation with a Xeon 2.66 GHz CPU and 4 GB RAM

Models	# vertices	Average time (s)	Max time (s)
Two Kids (Fig. 7)	49,996	0.029	0.082
Armadillo (Fig. 7)	172,974	0.215	0.875
Gargoyle (Fig. 7)	349,998	0.517	2.125
Feline (Fig. 7)	404,998	0.688	3.703
Horse (Fig. 7)	48,484	0.035	0.090
Beethoven (Fig. 7)	10,406	0.010	0.034
Oni (Fig. 7)	22,807	0.042	0.109
Decocube (Fig. 7)	19,992	0.008	0.068
Dog (Fig. 7)	75,002	0.221	0.594
Squirrel (Fig. 7)	13,280	0.021	0.044
Bunny (Fig. 7)	72,027	0.117	0.328
Lucy (Fig. 1)	50,002	0.019	0.054
Fertility (Fig. 11(a))	2,000	0.002	0.007
Fertility (Fig. 11(b))	5,000	0.004	0.015
Fertility (Fig. 11(c))	10,000	0.018	0.043
Dragon (Fig. 8(a))	5,000	0.004	0.028
Dragon (Fig. 8(b))	10,000	0.011	0.035
Dragon (Fig. 8(c))	50,000	0.031	0.088
Dragon (Fig. 8(d))	750,000	0.537	3.320

5.3 Topological Change

Our method can also efficiently handle topological changes during the curve evolution. Taking the dumbbell model shown in Fig. 9 as an example, the initial loop in Fig. 9a is split into two loops in Fig. 9b and further deformed to two degenerate points in the end, see Figs. 9c, 9d, 9e, 9f, and 9g. Technically, we achieve this in this way: if the $\mathring{\mathcal{E}}$-restricted geodesic loop passes through a vertex v twice or more, we can split $\mathring{\mathcal{E}}$ at v into two loops and then separately deform each loop into a geodesic loop or a degenerate point.

5.4 Robustness

As shown in Fig. 11, we test our method on the same 3D model with different resolutions and triangulations. The initial loops are specified by the same cutting plane. Our algorithm is stable and can produce consistent resultant geodesic loops for all of them.

5.5 Surfaces Embedded in High Dimension

We would like to further highlight that our proposed geodesic loop algorithm works for surfaces embedded in arbitrary-dimensional space, as the geodesic loop computation is intrinsic and independent of the embedding space.

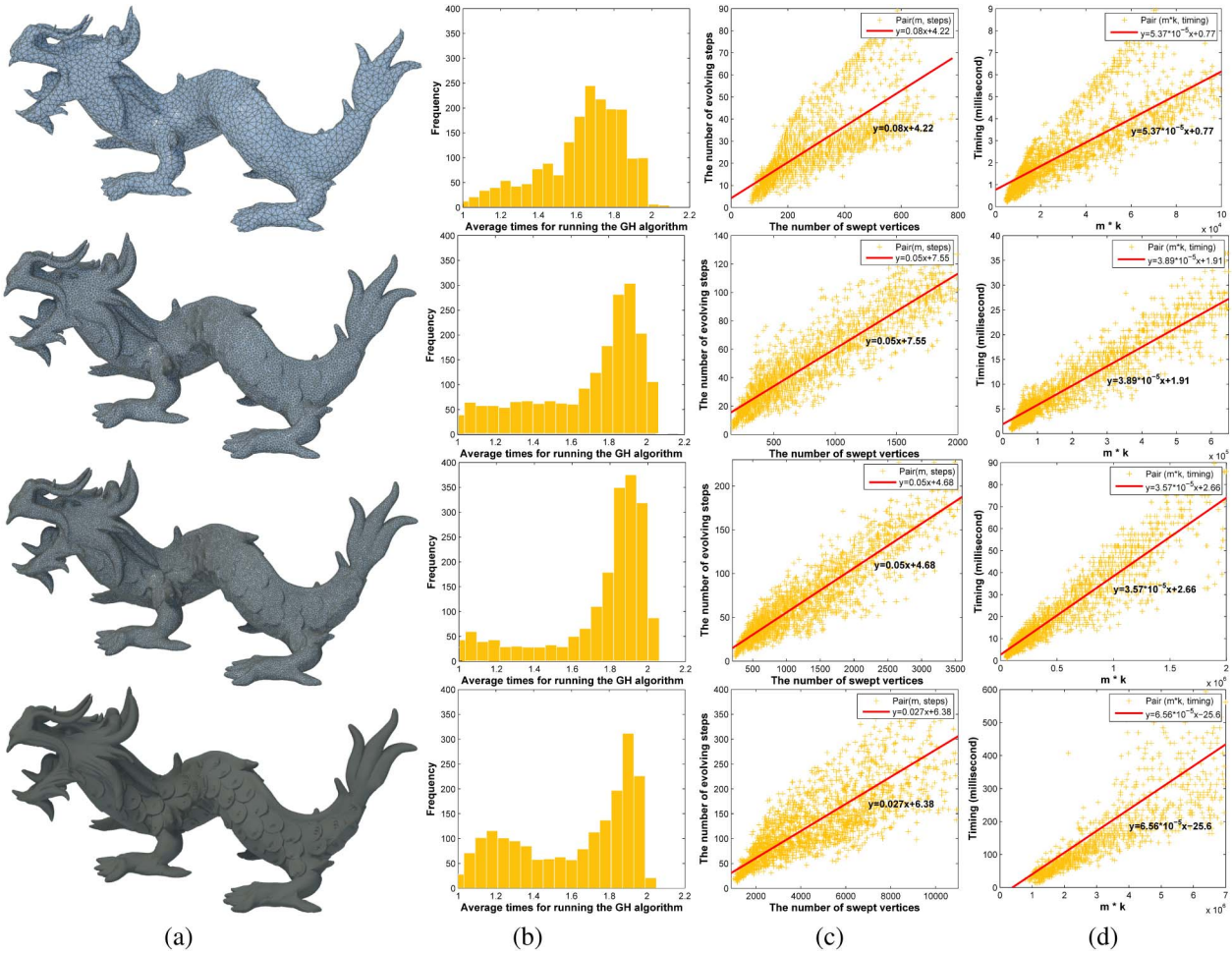


Fig. 8. Experimental average-time complexity analysis. Column (a) shows four Dragon models with different resolutions: $5K$, $10K$, $50K$, and $750K$ vertices. Column (b) shows a histogram of average number of times to run Guibas and Hershberger's algorithm [35] for computing a restricted shortest loop; the average is no more than 2. Column (c) shows that the number of evolving steps increases roughly linearly to the number of swept vertices. Column (d) reveals that the running time linearly depends on mk, where m is the number of vertices in the region bounded by the initial loop and the resultant geodesic loop, and k is the average number of edges in the edge sequences that the evolving loop passes through. The red lines in Columns (c) and (d) are fitted by the least-squares method.

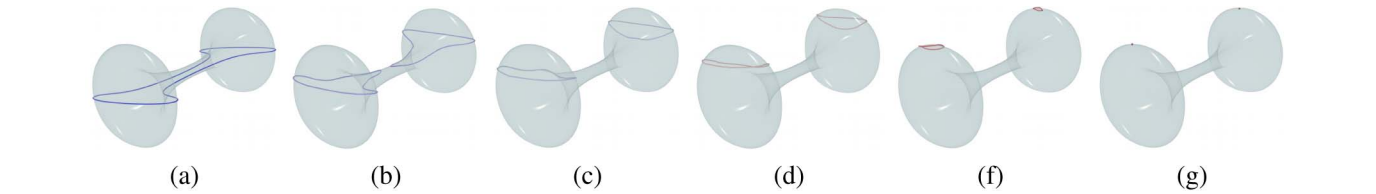


Fig. 9. Our algorithm can efficiently handle topological changes during the curve evolution. The initial loop (see (a)) is split into two loops (see (b)-(f) for intermediate results), both of which will eventually shrink to points (see (g)).

As shown in Fig. 12, given a surface in $\mathbb{R}^4$, we first compute the edge length using the vertex coordinates, and then apply our geodesic loop algorithm in which the curve evolution only requires the edge length and does not depend on the ambient space. Finally, once the geodesic loop is found, we compute the coordinate of every vertex on the polyline and then display the curve.

6 Discussions

Gu and his co-workers [17], [30] presented several techniques to compute the shortest loop for each homotopy class of oriented closed surface of high genus. Their methods require computing the uniformization metric and then embed the surface into a universal covering space. In this way, the uniformization metric can be computed by discrete Ricci flow [37] or Yamabe flow [31]. But since both curvature flows are highly nonlinear PDEs, the computational costs are very high. It is known that embedding surfaces of genus $g \geq 2$ into the hyperbolic space $\mathbb{H}^2$ could be error prone due to the numerical truncation during the computation. Therefore, the input surface meshes must come with good quality (i.e., the vertices are distributed uniformly over the surface) and contain a small number of vertices (usually a few thousands). Compared to this approach, our method is computationally efficient and numerically stable, and it can work for large-scale models as demonstrated in Section 5. Note that the computed geodesic using [30] is based on the uniformization metric rather than the original euclidean metric; hence, the resultant geodesics may not be exact. Here, we would like to also mention that our method can be easily applied to compute the geodesic homotopy basis as in [17] and [30]: by picking an arbitrary vertex v on a closed surface of genus g, one can compute the system of loops of v using [38], which contains $2g$ closed curves meeting at v; then, one can apply our discrete geodesic loop algorithm to evolve each curve into a v-relaxed geodesic (i.e., the curve is geodesic everywhere except at v), and obtain the resultant $2g$ curves that form the geodesic homotopy basis with base point v.

Wu and Tai [19] presented an algorithm to compute the geodesic loop on polygonal meshes using discrete geodesic curvature flow. They first converted the polygonal mesh to a level set, formulated geodesic curvature flow equation based on an energy minimization of curves, and then discretized the equation by finite volume method. By taking

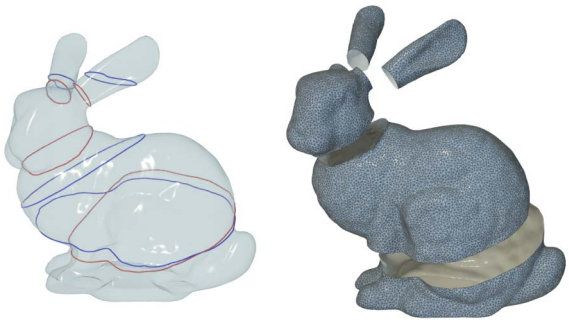


Fig. 10. Interactive shape segmentation. The user sketches on the 3D model to specify cuts; our algorithm then efficiently computes stable geodesic loops, along which the model is segmented.

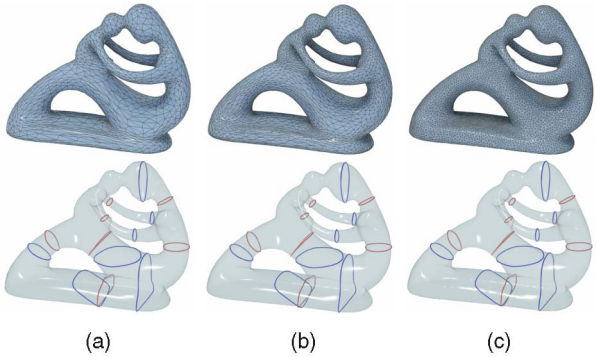


Fig. 11. Our algorithm is intrinsic to the geometry and insensitive to the triangulation/resolution. The same set of initial cutting planes applied to models with different resolutions result in sets of highly consistent geodesic loops. The initial loops and the resulting geodesic loops are drawn in blue and red, respectively. (a) 2K vertices. (b) 5K vertices. (c) 10K vertices.

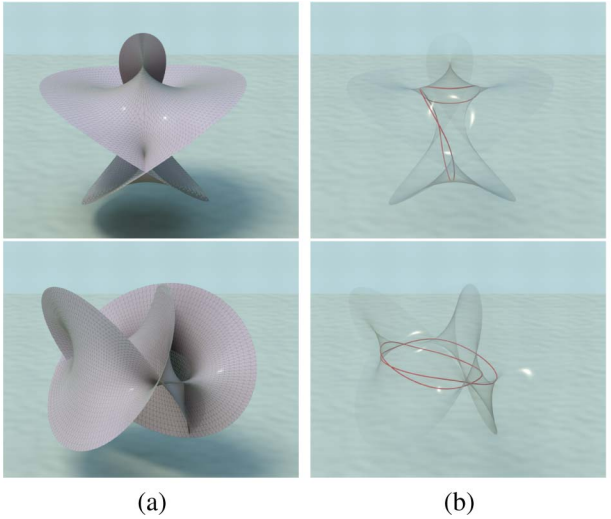


Fig. 12. Our geodesic loop algorithm is intrinsic and independent of the embedding space. Thus, it can be applied to surfaces embedded in higher dimensional space. (a) A genus-1 open surface in $\mathbb{R}^4$. (b) Two geodesic loops on it. Note that we have to project these results to 3-space for visualization. The top and bottom rows show two different 4D-to-3D projections, XYZ and XYW, respectively.

advantage of the implicit representation, Wu and Tai's algorithm is very flexible and effective to solving geodesic loop on weighted graphs. The geodesic curvature flow is, however, an iterative algorithm that converges to the exact solution only when the evolution time goes to ∞. Thus, the algorithm could have an early stop when the tolerance is less than a user-specified threshold. In other words, the geodesic curvature flow can lead only to *approximate* geodesic loops. By sharp contrast, our proposed method applies to the triangular meshes directly and can compute the *exact* geodesic loop within finite number of iterations. Furthermore, the performance of Wu and Tai's algorithm closely depends on the geometry of the surface and the position of the initial curve. If the initial curve is not close to the geodesic, it takes longer time (in the order of *minutes* for a medium-sized mesh as reported in [19]) to converge. Later on, by reducing the problem dimension, Zhang et al. [6] proposed fast geodesic curvature flow that is much faster (in the order of *seconds*) than dGCF. Note that both dGCF and FGCF heavily depend on a numerical solver of a linear system in the curve evolution. By contrast, our method does not require any numerical package, thus, is more efficient, robust, and easy to implement. As shown in Table 1, our method converges extremely fast (in the order of *milliseconds* for meshes of similar size) even if the initial loop is far from the optimal solution.

Similar to [6], our method also has the potential for interactive shape segmentation due to its high performance. As shown in Fig. 10, the users can freely sketch a cutting plane on the 3D model; our algorithm then deforms the initial planar cutting curve into a geodesic loop, along which the shape is segmented. However, we noticed that a good segmentation reflecting geometric features and human shape perception may not be a geodesic, e.g., the curve enclosing Bunny's tail is not a geodesic. One possible way to extend our algorithm for mesh segmentation is to compute a feature-aware metric such that the geodesic loop under this new metric can enclose the user-desired feature. We will investigate into this direction in our future work.

7 Conclusions

This paper proved the existence and uniqueness of geodesic loops restricted on a closed edge sequence. Based on this result, we devised an efficient method to compute exact geodesic loops on triangular meshes. Taking an arbitrary loop as input, our method can efficiently evolve it into an exact geodesic loop. In contrast to existing methods for computing geodesic loops, our method is exact, efficient, intrinsic, stable, robust, and insensitive to mesh triangulations and resolutions. Hence, we are able to apply it to interactive applications such as the interactive shape segmentation example demonstrated in this paper.

Currently, we proved that the shortest loop in a closed edge sequence with k edges can be computed in the best case $O(k)$ time and the worst case $O(k^2)$ time. Based on our extensive experiments on a wide range of meshes, we observe that our algorithm can usually run at a speed much faster than the suggested worst case scenario. Thus, we conjecture that the average case is still in $O(k)$ time.

Acknowledgments

The authors would like to thank the anonymous reviewers for their careful reviews and constructive comments. Special thanks to Ms. Shuang-Min Chen for her help on preparing the figures and video demonstration. This project was partially supported by NRF2008IDM-IDM-004-006, AcRF Tier1 69/07, and AcRF Tier1 RG 13/08. Correspondence should be addressed to Y. He (yhe@ntu.edu.sg).

References

[1] H. Lee, Y. Tong, and M. Desbrun, "Geodesics-Based One-to-One Parameterization of 3D Triangle Meshes," *IEEE Multimedia*, vol. 12, no. 1, pp. 27-33, Jan. 2005.

[2] G. Peyré and L.D. Cohen, "Geodesic Remeshing Using Front Propagation," *Int'l J. Computer Vision*, vol. 69, no. 1, pp. 145-156, 2006.

[3] Q.-X. Huang, B. Adams, M. Wicke, and L.J. Guibas, "Non-Rigid Registration under Isometric Deformations," *SGP '08: Proc. Sixth Symp. Geometry Processing*, pp. 1449-1457, 2008.

[4] P.V. Sander, Z.J. Wood, S.J. Gortler, J. Snyder, and H. Hoppe, "Multi-Chart Geometry Images," *SGP '03: Proc. Eurographics/ACM SIGGRAPH Symp. Geometry Processing*, pp. 146-155, 2003.

[5] K. Zhou, J. Synder, B. Guo, and H.-Y. Shum, "Iso-Charts: Stretch-driven Mesh Parameterization Using Spectral Analysis," *SGP '04: Proc. Eurographics/ACM SIGGRAPH Symp. Geometry Processing*, pp. 45-54, 2004.

[6] J. Zhang, C. Wu, J. Cai, J. Zheng, and X.-C. Tai, "Mesh Snapping: Robust Interactive Mesh Cutting Using Fast Geodesic Curvature Flow," *Computer Graphics Forum*, vol. 29, no. 2, pp. 517-526, 2010.

[7] M. Sharir and A. Schorr, "On Shortest Paths in Polyhedral Spaces," *SIAM J. Computing*, vol. 15, no. 1, pp. 193-215, 1986.

[8] J.S.B. Mitchell, D.M. Mount, and C.H. Papadimitriou, "The Discrete Geodesic Problem," *SIAM J. Computing*, vol. 16, no. 4, pp. 647-668, 1987.

[9] S.-Q. Xin and G.-J. Wang, "Improving Chen and Han's Algorithm on the Discrete Geodesic Problem," *ACM Trans. Graphics*, vol. 28, no. 4, pp. 1-8, 2009.

[10] R. Bott, "On the Iteration of Closed Geodesics and the Sturm Intersection Theory," *Comm. Pure and Applied Math.*, vol. 9, no. 2, pp. 171-206, 1956.

[11] M. Freedman, J. Hass, and P. Scott, "Closed Geodesics on Surfaces," *Bull. London Math. Soc.*, vol. 14, pp. 385-391, 1982.

[12] M. Jin, F. Luo, S.-T. Yau, and X. Gu, "Computing Geodesic Spectra of Surfaces," *Proc. ACM Symp. Solid and Physical Modeling*, pp. 387-393, 2007.

[13] F. Hétroy and D. Attali, "From a Closed Piecewise Geodesic to a Constriction on a Closed Triangulated Surface," *PG '03: Proc. 11th Pacific Conf. Computer Graphics and Applications*, p. 394, 2003.

[14] D. Reniers and A. Telea, "Part-Type Segmentation of Articulated Voxel-Shapes Using the Junction Rule," *Proc. Pacific Graphics*, 2008.

[15] M. Dixon, N. Jacobs, and R. Pless, "Finding Minimal Parameterizations of Cylindrical Image Manifolds," *CVPRW '06: Proc. Conf. Computer Vision and Pattern Recognition Workshop*, p. 192, 2006.

[16] T.K. Dey and S. Guha, "Transforming Curves on Surfaces," *J. Computer and System Sciences*, vol. 58, pp. 297-325, 1999.

[17] X. Yin, M. Jin, and X. Gu, "Computing Shortest Cycles Using Universal Covering Space," *The Visual Computer*, vol. 23, no. 12, pp. 999-1004, 2007.

[18] L. Fan, L. Liu, and K. Liu, "Paint Mesh Cutting," *Computer Graphics Forum*, vol. 30, no. 2, pp. 603-612, 2011.

[19] C. Wu and X. Tai, "A Level Set Formulation of Geodesic Curvature Flow on Simplicial Surfaces," *IEEE Trans. Visualization and Computer Graphics*, vol. 16, no. 4, pp. 647-662, July/Aug. 2010.

[20] J. Chen and Y. Han, "Shortest Paths on a Polyhedron," *SCG '90: Proc. Sixth Ann. Symp. Computational Geometry*, pp. 360-369, 1990.

[21] V. Surazhsky, T. Surazhsky, D. Kirsanov, S.J. Gortler, and H. Hoppe, "Fast Exact and Approximate Geodesics on Meshes," *ACM Trans. Graphics*, vol. 24, no. 3, pp. 553-560, 2005.

[22] K. Polthier and M. Schmies, "Straightest Geodesics on Polyhedral Surfaces," *Mathematical Visualization*, p. 391, Springer Verlag, 1998.

[23] K. Polthier and M. Schmies, "Geodesic Flow on Polyhedral Surfaces," *Proc. Eurographics-IEEE Symp. Scientific Visualization*, pp. 179-188, 1999.

[24] S.-Q. Xin and G.-J. Wang, "Efficiently Determining a Locally Exact Shortest Path on Polyhedral Surfaces," *Computer-Aided Design*, vol. 39, no. 12, pp. 1081-1090, 2007.
[25] R. Kimmel and J.A. Sethian, "Computing Geodesic Paths on Manifolds," *Proc. Nat'l Academy of Sciences of USA*, vol. 95, pp. 8431-8435, 1998.
[26] S. Har-Peled, "Approximate Shortest paths and Geodesic Diameter on a Convex Polytope in Three Dimensions," *Discrete and Computational Geometry*, vol. 21, no. 2, pp. 217-231, 1999.
[27] P.K. Agarwal, S. Har-Peled, and M. Karia, "Computing Approximate Shortest Paths on Convex Polytopes," *Proc. Symp. Computational Geometry*, pp. 270-279, 2000.
[28] A. Spira and R. Kimmel, "Geodesic Curvature Flow on Parametric Surfaces," *Proc. Curve and Surface Design*, pp. 365-373, 2002.
[29] P. Diaz-Gutierrez, D. Eppstein, and M. Gopi, "Curvature Aware Fundamental Cycles," *Computer Graphics Forum*, vol. 28, no. 7, pp. 2015-2024, 2009.
[30] W. Zeng, M. Jin, F. Luo, and X. Gu, "Canonical Homotopy Class Representative Using Hyperbolic Structure," *Proc. IEEE Int'l Conf. Shape Modeling and Applications*, 2009.
[31] W. Zeng, Y. He, J. Xia, X. Gu, and H. Qin, "C^{∞} Smooth Freeform Surfaces over Hyperbolic Domains," *SPM '09: Proc. SIAM/ACM Joint Conf. Geometric and Physical Modeling*, pp. 367-372, 2009.
[32] J. Zhang, J. Zheng, and J. Cai, "Interactive Mesh Cutting Using Constrained Random Walks," *IEEE Trans. Visualization and Computer Graphics*, vol. 17, no. 3, pp. 357-367, Mar. 2011.
[33] Y. Zheng and C.-L. Tai, "Mesh Decomposition with Cross-Boundary Brushes," *Computer Graphics Forum*, vol. 29, no. 2, pp. 527-535, 2010.
[34] A. Golovinskiy and T. Funkhouser, "Randomized Cuts for 3D Mesh Analysis," *Proc. ACM SIGGRAPH ASIA Papers*, vol. 27, Dec. 2008.
[35] L.J. Guibas and J. Hershberger, "Optimal Shortest Path Queries in a Simple Polygon," *SCG '87: Proc. Third Ann. Symp. Computational Geometry*, pp. 50-63, 1987.
[36] S. Kakutani, "Ein Beweis des sätzes Von Edelheit über Konvexe Mengen," *Proc. Imp. Acad. Tokyo 13*, pp. 93-94, 1937.
[37] M. Jin, F. Luo, and X.D. Gu, "Computing General Geometric Structures on Surfaces Using Ricci Flow," *Computer-Aided Design*, vol. 39, no. 8, pp. 663-675, 2007.
[38] E.C. de Verdière and F. Lazarus, "Optimal System of Loops on an Orientable Surface," *Proc. 43rd Symp. Foundations of Computer Science*, pp. 627-636, 2002.

Shi-Qing Xin received the PhD degree in applied mathematics from Zhejiang University in 2009. He had a two-year work experience at Autodesk, Shanghai. Since 2009, he has been working at Nanyang Technological University as a research fellow. He focused on computing discrete geodesics in the past and his research interests include computational geometry, computer graphics, and topology analysis.

Ying He received the BS and MS degrees in electrical engineering from Tsinghua University, China, and the PhD degree in computer science from the State University of New York (SUNY), Stony Brook. He is currently an assistant professor at the School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include computer graphics, computer-aided design, and scientific visualization. His major works include manifold splines, polycube parameterization, volumetric mapping, and computer generated line drawings. More information about his research can be found at http://www.ntu.edu.sg/home/yhe. He is a member of the IEEE.

Chi-Wing Fu received the BSci and MPhil degrees in computer science and engineering from the Chinese University of Hong Kong, in 1997 and 1999, respectively, and the PhD degree in computer science from Indiana University Bloomington in 2003. He is currently an assistant professor in the School of Computer Engineering at Nanyang Technological University, Singapore. His research interests include tile-based modeling and rendering, recreational graphics, astrophysical and mathematical visualization, and user interaction. He received the IEEE Transactions on Multimedia Prize Paper Award in 2005. He has been a member of the IEEE since 1997.