Towards Geodesic Ridge Curve for Region-wise Linear Representation of Geodesic Distance Field*

Wei Liu[a,1], Pengfei Wang[a,1], Shuangmin Chen[b,*], Shiqing Xin[a], Changhe Tu[a], Ying He[c] and Wenping Wang[d]

[a]Shandong University, China
[b]Qingdao University of Science and Technology, China
[c]Nanyang Technological University, Singapore
[d]Texas A&M University, USA

ARTICLE INFO

Keywords:
Digital Geometry Processing
Geodesic Distance Field
Ridge Point
Geodesic Isoline

ABSTRACT

This paper addresses the challenge of representing geodesic distance fields on triangular meshes in a piecewise linear manner. Unlike general scalar fields, which often assume piecewise linear changes within each triangle, geodesic distance fields pose a unique difficulty due to their non-differentiability at ridge points, where multiple shortest paths may exist. An interesting observation is that the geodesic distance field exhibits an approximately linear change if each triangle is further decomposed into sub-regions by the ridge curve. However, computing the geodesic ridge curve is notoriously difficult. Even when using exact algorithms to infer the ridge curve, desirable results may not be achieved, akin to the well-known medial-axis problem. In this paper, we propose a two-stage algorithm. In the first stage, we employ Dijkstra's algorithm to cut the surface open along the dual structure of the shortest path tree. This operation allows us to extend the surface outward (resembling a double cover but with distinctions), enabling the discovery of longer geodesic paths in the extended surface. In the second stage, any mature geodesic solver, whether exact or approximate, can be employed to predict the real ridge curve. Assuming the fast marching method is used as the solver, despite its limitation of having a single marching direction in a triangle, our extended surface contains multiple copies of each triangle, allowing various geodesic paths to enter the triangle and facilitating ridge curve computation. We further introduce a simple yet effective filtering mechanism to rigorously ensure the connectivity of the output ridge curve. Due to its merits, including robustness and compatibility with any geodesic solver, our algorithm holds great potential for a wide range of applications. We demonstrate its utility in accurate geodesic distance querying and high-fidelity visualization of geodesic iso-lines.

1. Introduction

Given a 3D model $\mathcal{S}$ and a base point s on its surface, the geodesic ridges with respect to s denote the set of points where multiple geodesic shortest paths exist from s. These ridge points typically form a curved cut locus, along which the surface can be split into a topological disk. The ridge curve is crucial in characterizing the geometric and topological structure (Sakai, 1996).

From a differential geometry perspective, the geodesic distance function is non-differentiable at geodesic ridge points, distinguishing it from a smooth scalar field on the surface. While a smooth scalar field can be adequately approximated by assuming linearity within each triangle, especially for small triangles, the geodesic distance field cannot be simply represented as piecewise linear due to non-differentiability. This departure from linearity introduces inaccuracies in applications such as geodesic distance estimation near the ridge curve and the extraction of geodesic isolines. Building a piecewise linear representation of a geodesic distance field is imperative.

We observe that the geodesic distance field is approximately linear within a triangle when further decomposing the triangle into small sub-regions along ridge curves. This observation motivates our paper's theme: devising a robust

*This work is supported by National Key R&D Program of China (2022YFB3303200), National Natural Science Foundation of China (62272277, U23A20312, 62072284) and NSF of Shandong Province (ZR2020MF036).

*Corresponding author
202100130071@mail.sdu.edu.cn (W. Liu); pengfei1998@foxmail.com (P. Wang); csmqq@163.com (S. Chen)
ORCID(s): 0009-0004-7554-1730 (W. Liu); 0000-0002-2079-275X (P. Wang)

[1]These authors contributed to the work eqully and should be regarded as co-first authors.

algorithm to compute the geodesic ridge curve. To the best of our knowledge, computing the geodesic ridge curve is notoriously difficult, with few available algorithms. The most recent algorithm, proposed by Mancinelli et al. (2021), includes a filtering step using a gradient-norm tolerance, making it less robust. Tests reveal that the reported ridge curve exhibits zigzag patterns, deviating significantly from the actual ridge curve. Moreover, it appears that exact geodesic algorithms can report the ridge curve during the propagation of windows, but this is misleading for two reasons. First, tracing the ridge curve during the execution of exact geodesic algorithms is challenging due to potential hyperbolic segments existing in the real ridge curve. Second, the ridge curve reported by window propagation stretches across the surface and is highly dependent on the triangulation. The desirable ridge curve should reveal the underlying geometry rather than being influenced by the triangulation.

In this paper, we propose a two-stage algorithm. In the first stage, we employ Dijkstra's algorithm to obtain the shortest path tree and its dual structure. We identify non-trivial cycles and cut the surface into topological disks. Subsequently, we extend the surface outward, resembling a double cover but with distinctions, enabling each triangle to have multiple copies in the extended surface. This extension allows for the survival of long geodesic paths, even if they are not the shortest. In the second stage, we use a geodesic algorithm to refine the cut locus, ensuring that it accurately reflects how geodesic paths converge from different directions. Our algorithm allows any geodesic solver to drive the computation of geodesic distances. For example, the fast marching method assumes the linearity of the distance field and has a single marching direction in a triangle, but our extended surface, containing multiple copies of each triangle, enables geodesic paths to enter the triangle from various directions. Furthermore, we introduce a simple yet effective filtering technique to rigorously ensure the connectivity of the output ridge curve. In Figure 1, two examples of ridge-curve computed by our algorithm are presented, with the fast marching method employed as the geodesic solver. The remaining displayed results in the paper use the VTP algorithm (Qin et al., 2016) as the geodesic solver. We demonstrate its utility in accurate geodesic distance querying and high-fidelity visualization of geodesic iso-lines.

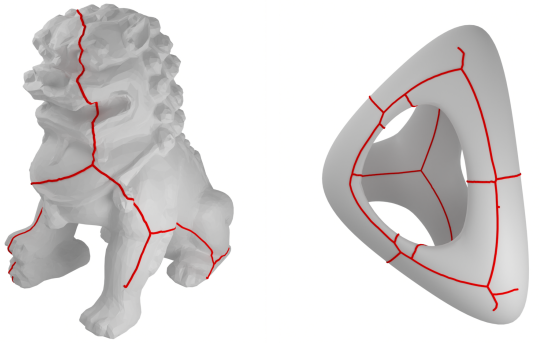


Figure 1: Ridge curve examples computed by our algorithm.

2. Related Works

2.1. Topological cut locus

The concept of a cut locus was introduced by Poincaré (1905). Consider a compact real-analytic Riemannian 2-manifold denoted as M, essentially a closed surface equipped with a smooth Riemannian metric. The typical cut locus of a base point s in M is defined as the set of points where minimizing geodesics issued from s stop being minimizing. Due to its advantageous property of cutting the surface into a topological disk (Mancinelli et al., 2021), the cut locus is essential for analyzing geometric and topological structures. For instance, Myers (1935) established, based on topological considerations, that the ridges of a closed real-analytic surface form a graph with a finite number of branches. Buchner (1977) extended this result to higher dimensions, demonstrating that in dimension two, the local structure resembles that of a tree. Sakai (1996) proved that the cut locus shares the same homology as M, i.e., it forms a tree for surfaces with genus zero but contains $2g$ cycles for high-genus surfaces, where g is the genus of M.

2.2. Graph-based non-trivial cycles

Direct computation of the cut locus for a smooth surface is not straightforward. In past research, most researchers initially extract a graph embedded on the surface and then aim to find the shortest non-trivial cycles. Cutting a surface to reduce its topological complexity is a common technique used in geometric computing and topological graph theory. Erickson and Har-Peled (2002) discuss the relevance of cutting a surface to obtain a topological disk in computer graphics. Thomassen (1990) was the first to provide a polynomial-time algorithm for finding a shortest non-separating and a shortest non-contractible cycle in a graph on a surface. Although Thomassen does not claim any specific running time, his algorithm attempts a quadratic number of cycles, and for each one, it has to determine if it is non-separating or non-contractible. This yields a rough estimate of $O(n(n+g)^2)$ for its running time, where n denotes the complexity of the surface. Rote (2003) describes applications that algorithmic problems involving curves on topological surfaces have in other fields. Kutz (2006) presented an algorithm that computes a shortest non-contractible and a shortest non-separating cycle on an orientable combinatorial surface of bounded genus in $O(n \log n)$ time.

2.3. Geodesic algorithms

Given a surface with an analytic representation, the query of a geodesic path can be formulated as the Euler-Lagrange partial differential equation in differential geometry. However, in general, the PDE has no closed-form solution, necessitating the search for a numerical solution. The discrete geodesic problem typically takes a triangle mesh as the input and aims to find the shortest paths represented as polylines on the polygonal surface.

Exact geodesic algorithms. Representative exact algorithms include the MMP algorithm (Mitchell et al., 1987), the CH algorithm (Chen and Han, 1990), and some variant versions (Surazhsky et al., 2005; Xin and Wang, 2009; Liu, 2013; Xu et al., 2015; Qin et al., 2016). These algorithms commonly employ a window to encode geodesic paths that share the same edge sequence, enabling the representation of infinitely many geodesic paths with a finite number of discrete windows. The key idea is to propagate a dynamic outward wavefront across mesh faces in a Dijkstra-like sweep.

Exact geodesic algorithms cannot be directly applied to find ridge curves. The main reason is that the ridge curve reported by window propagation stretches across the surface and is highly dependent on the triangulation. However, the desirable ridge curve should be sufficiently clean to reveal the underlying geometry, rather than being influenced by the triangulation.

Approximate geodesic algorithms. There are approximate geodesic distance computation algorithms like the Fast Marching method (Kimmel and Sethian, 1998) and the heat method (Crane et al., 2013). For example, the heat method (Crane et al., 2013), as well as its parallel implementation (Tao et al., 2019), utilizes an observation that the heat field and the geodesic distance field share the common isolines (but may have different values). They need to normalize the gradient field into a unit vector field before the reconstruction of the distance field. These methods provide efficient approximations of geodesic distances on the model.

2.4. Challenges in geodesic ridge curve computation

Sinclair and Tanaka (2002) computes a piecewise polynomial approximation to the exponential map and numerically inverts it, accurately considering the global nature of the geodesic ridges. However, it is limited to handling models with a genus of 1. Itoh and Sinclair (2004) addresses the issue of the cut locus becoming dense in the smooth surface in the limit of infinitely small triangles by introducing a minimal angular resolution. However, it is important to note that the proposed method supports only convex surfaces. Misztal et al. (2011) observes the process of forward propagation of the distance field, detecting the cut locus from self-intersections of the propagated front. Theoretically, the method is applicable to surfaces of arbitrary genus. However, the computational cost is extremely high, and it suffers from numerical issues. Mancinelli et al. (2021) utilizes the computation of triangle gradient values to estimate the positions of ridges, ensuring correct ridge topology through post-processing. The method demonstrates a remarkably fast computational speed. However, due to the approximate nature of gradients, the estimated ridges often lack accuracy, particularly in cases of larger triangles or lower triangulation quality in the model. Tests also show that its post-processing step is not robust, leading to frequent failures in complex scenarios. Générau et al. (2022) provides a thorough examination of the topic, introducing a rigorous mathematical approach to compute an approximation of the cut locus. Nevertheless, achieving a practical and robust algorithm based on Générau et al. (2022) appears to be a non-trivial task.

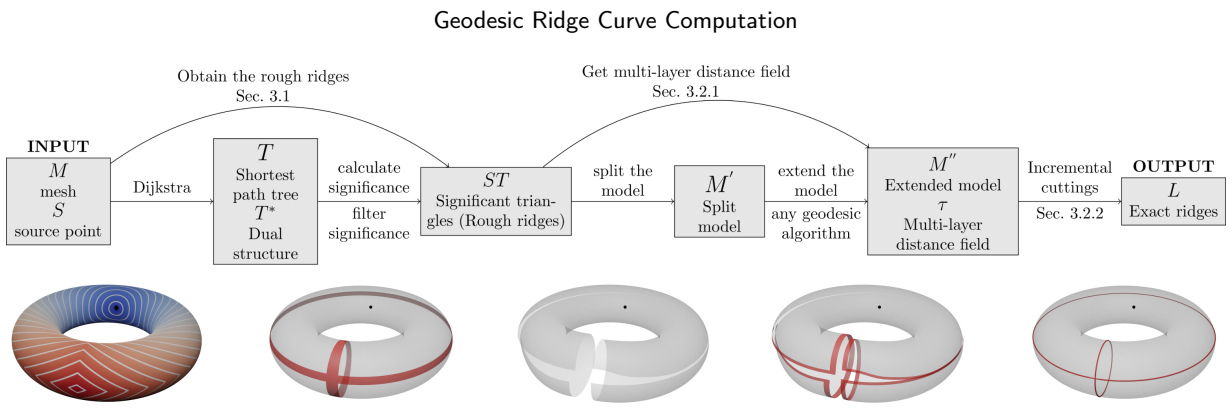


Figure 2: The pipeline of our algorithm.

3. Algorithm

Our algorithm consists of two stages for computing accurate and smooth geodesic ridges on the given surface. Initially, we employ Dijkstra's algorithm for computation, cutting the surface into a topological disk. For genus-0 models, the model can be directly cut along the dual of the shortest path tree. However, for higher-genus models, identifying non-trivial cycles is necessary, aided by both the shortest path tree and its dual. It's important to note that the cut locus in this stage follows mesh edges, resulting in a zigzag pattern. However, this characteristic does not have any adverse effect on the final ridge curve. In the next stage, we extend the surface to generate sufficient copies of each triangle, facilitating a sufficiently large number of geodesic paths crucial for determining the actual ridge curve. Subsequently, we run an arbitrary geodesic solver (provided the result is sufficiently accurate) on the extended surface. Finally, the ridge curve within a triangle is obtained through an incremental hyperplane cutting operation, similar to the approach demonstrated in Xin et al. (2022). Refer to Fig. 2 for an illustration of the procedure. We shall detail each stage in the following subsections.

3.1. Rough ridge curve

As discussed in Section 2, several graph-based algorithms aim to find the initial ridge curve by treating the input polygonal surface as a graph. In our approach, we adopt the strategy proposed by Erickson and Whittlesey (2005) for this purpose. By implementing Dijkstra's algorithm, we obtain a shortest path tree that follows the mesh edges. One can take its dual as the initial ridge curve, assuming the input model has a genus of zero. For models with higher genera, an additional step is required to extract non-trivial cycles. (Details will be elaborated later.)

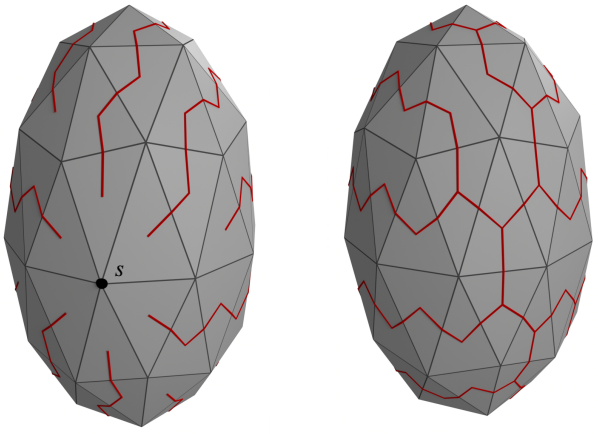
However, it's important to note that the initial ridge curve is distinct from the target ridge curve. The main difference lies in the fact that the graph-based ridge curve is distributed over the triangular surface and closely tied to the triangulation. In contrast, the target ridge curve we aim to compute should be independent of triangulation and capable of revealing the underlying geometric variations. Therefore, to obtain the significant portion, we need to carefully invent a pruning mechanism that preserves connectivity.

Genus-0 models. Consider a mesh edge v_1v_2. If either v_1 provides the shortest distance to v_2 or vice versa, we designate v_1v_2 as a segment of the shortest path tree T. Let's assume that the two faces sharing v_1v_2 are f_i and f_j. If $v_1v_2 \notin T$, then f_i and f_j form a pair of "contiguous" triangles. The dual of T, denoted as T^*, can be defined by enumerating all such "contiguous" triangles. See Figure 3 for an illustration. In the following, we discuss the significance of a pair of "contiguous" triangles. We define the significance of v_1v_2 as the area of all the triangles enclosed by $\pi(s, v_1) \cup \pi(s, v_2) \cup v_1v_2$, where $\pi(\cdot, \cdot)$ denote the operation of taking the shortest path.

To be more precise, $\pi(s, v_1) \cup \pi(s, v_2) \cup v_1v_2$ split the genus-0 surface into parts, we denote the two areas by $\text{Area}^+_{v_1v_2}$ and $\text{Area}^-_{v_1v_2}$. The significance of v_1v_2 is given by the smaller one, i.e.,

$$I(v_1v_2) = \min(\text{Area}^+_{v_1v_2}, \text{Area}^-_{v_1v_2}).$$

In the following, we shall explain why the connectivity of the ridge curve remains unchanged for any tolerance. Take Figure 4 for an example. We assume that by choosing some tolerance, the edge v_1v_2 has been removed from T^*, making



(a) Front view (b) Back view

Figure 3: The source point is denoted as s. By running Dijkstra's algorithm, we obtain the shortest path tree T. The dual structure, T^*, is visualized in red.

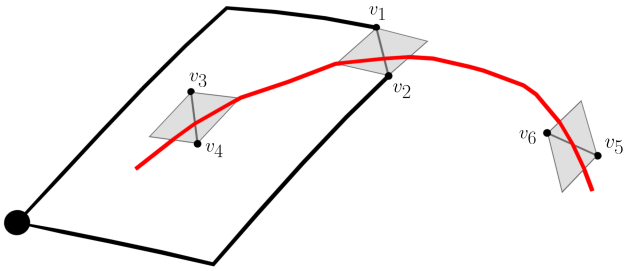


Figure 4: The thick black lines represent the shortest path, while the red lines indicate the ridge curve. $\pi(s, v_1) \cup \pi(s, v_2) \cup v_1v_2$ divides the genus-0 surface into segments with respective areas $\text{Area}^+_{v_1v_2}$ and $\text{Area}^-_{v_1v_2}$. To quantify the significance of v_1v_2, we define its importance as $I(v_1v_2) = \min(\text{Area}^+_{v_1v_2}, \text{Area}^-_{v_1v_2})$, ensuring that the connectivity of the ridge curve remains unchanged for any tolerance.

the ridge curve broken when it crosses v_1v_2. Since $\pi(s, v_1) \cup \pi(s, v_2) \cup v_1v_2$ divides the surface into two parts, the ridge curve still partially exists in both parts. Suppose that v_3v_4 is located in the portion that defines $I(v_1v_2)$. Then it is easy to know that

$$I(v_3v_4) < I(v_1v_2),$$

which contradicts the assumption that the ridge curve still partially exists in both parts. If $I(v_1v_2)$ is given by another part, then we have

$$I(v_5v_6) < I(v_1v_2),$$

which also leads to a contradiction. Therefore, while the definition of significance may seem straightforward, the most notable feature is that it ensures the ridge curve, even after pruning, remains a connected component. It's also worth noting that the maximum possible tolerance cannot exceed one-half of the area of the whole surface; otherwise, the ridge curve becomes empty.

Fig. 5 illustrates how we transition from the significance of the original model edges to the vertices of the T^* structure. Suppose that at each vertex i of the dual structure T^*, we define $\text{A}[i]$ to be the area of the triangle corresponding to this vertex. In our definition, every edge uv that is not in the shortest path structure T will always correspond to an edge $u'v'$ in the dual structure T^*. The $I(uv)$ is equivalent to the part of $\sigma_{u'}$ and $\sigma_{v'}$ that minimizes the sum of triangle face area values of all vertices in the two parts after $u'v'$ is disconnected in the dual structure, namely:

$$I(uv) = \min(\Sigma_{i\in\sigma_{u'}}\text{A}[i],\ \Sigma_{i\in\sigma_{v'}}\text{A}[i])$$

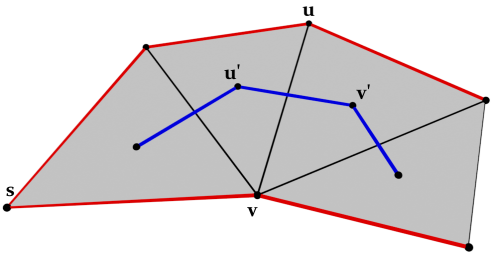


Figure 5: In the figure, we show that an edge uv on the original model that is not part of the shortest path can always correspond to an edge $u'v'$ on the dual structure T^*, and we define the connected branch of u' as $\sigma_{u'}$ and the connected branch of v' as $\sigma_{v'}$ after removing the edge $u'v'$ from the dual structure T^*.

Given the definition above, we understand that the significance of an edge is actually the accumulation of face area values from all vertices in the smaller of the two branches it separates. We define this value as Acc, as presented in Fig. 5, Acc$[u']$ represents the sum of triangle areas in branch denoted as $\sigma_{u'}$.

To get the accumulated areas Acc, we begin traversing T^* from the degree one vertices whose accumulation value is simply their corresponding face areas. If we encounter a vertex u with degree two, we record the accumulated value u as $\text{Acc}[u] = \min(\phi(u),\ \text{Area}_{total} - \phi(u))$ where $\phi(u) = \text{Acc}[v] + \text{A}[u]$, v is the previous vertex during traversal.

If we encounter a vertex w of degree three, we would like to store the value Acc$[v]$ in a map that can be used for latter reference. If one value Acc$[u]$ already exists in the corresponding map, the accumulation value of w is defined as $\min(\phi(w),\ \text{Area}_{total} - \phi(w))$ where $\phi(w) = \text{Acc}[u] + \text{Acc}[v] + \text{A}[w]$ and v is the previous vertex during traversal. Subsequently the traversal continues by considering the vertex w as a vertex of degree one. The algorithm terminates when all vertices have corresponding accumulate values assigned. Pseudocode for this part is provided in the appendix.

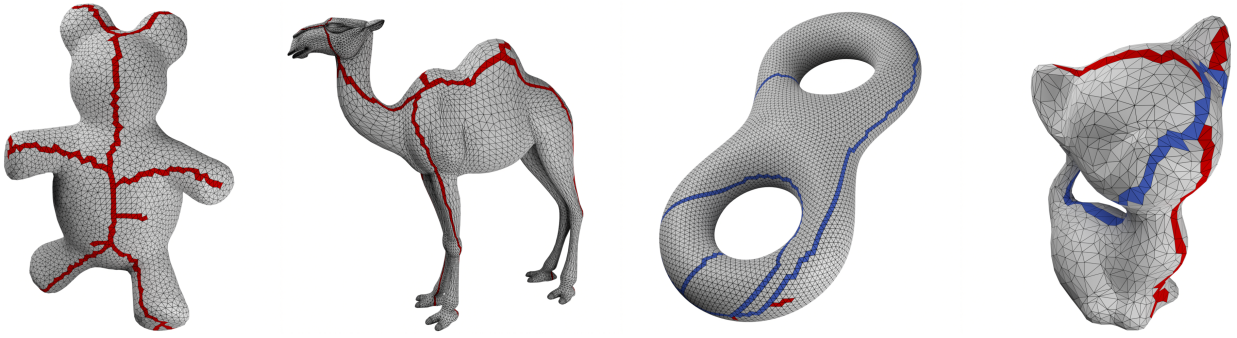


Figure 6: The two models on the left are of genus 0, while the two models on the right are high-genus. The blue triangles represent non-contractible topological cycles, while the red triangles depict those significant branches of T^* after being pruned by a certain tolerance.

Models of arbitrary genus. Suppose that E is the edge set of the input model. According to Erickson and Whittlesey (2005), for high-genus models, the following edge set is not empty:

$$\{e \mid e \notin T \ \&\& \ e \notin T^*\},$$

which defines the non-contractible topological cycles.

A significant distinction between genus-0 models and high-genus models is that non-contractible topological cycles cannot be removed from T^*. Doing so would result in an inconsistency between the topology of the ridge curve and the base surface. Let v_1v_2 be one mesh edge contributing to the non-contractible topological cycles (both the two faces incident to v_1v_2 are located on the non-contractible topological cycles; See Figure 6). In this situation, $\pi(s, v_1) \cup \pi(s, v_2) \cup v_1v_2$ cannot split the surface into two parts. Instead, we have

$$\text{Area}^+_{v_1v_2} = \text{Area}^-_{v_1v_2} = \text{Area}_{\text{total}}.$$

According to the definition of significance, the significance of v_1v_2 exactly equals the entire surface area. In fact, this holds true for each mesh edge contributing to the non-contractible topological cycles. This further implies that as long as the significance tolerance is less than the entire area, the non-contractible topological cycles persist.

3.2. Computing accurate geodesic ridges

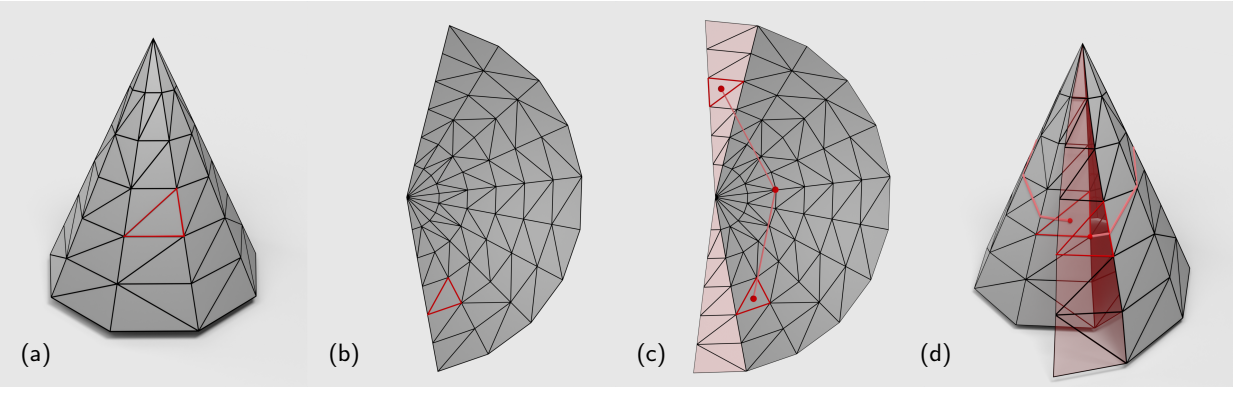


Figure 7: (a-b) We can cut the model apart along the cut locus. (c) shows a one-ring extension, where the red triangle produces two copies, allowing for geodesic paths to enter the triangle from two different directions. (d) An alternative visualization of the one-ring extended surface.

3.2.1. Extending the surface

Recall that although the cut locus is approximate, it roughly represents the points where geodesic distance stops increasing. To find the actual ridge points, we need to allow geodesic paths to enter a triangle from various possible directions. Therefore, we need to extend the surface from the cut locus. Suppose that $F^{(1)}$ is the triangles incident to the cut locus. We can add $F^{(1)}$ to the original face set F, which we call a *1-ring extension*. In this way, we can specify an arbitrary k, yielding the k-ring extended surface; See Figure 7 for an illustration. However, a too-large k is unnecessary. Indeed, surface extension is a dynamic process, and the initial selection of k is not as critical. Moving forward, we will provide more specific descriptions.

3.2.2. Geodesic distances on the extended surface

To achieve this, we can run any geodesic solver on the extended surface. Taking $\triangle v_1v_2v_3$ as an illustrative example, we use $\{\triangle^i v_1v_2v_3\}_{i=1}^m$ to denote its copies during the surface extension. After running the geodesic solver, each mesh vertex of the extended surface has the shortest distance to the source point. We assume that the change in the distance value within each triangle is linear. Note that different copies of the same triangle may have different linear distance fields.

Let d_1^i, d_2^i, d_3^i be the shortest distances at the three vertices of the i-th copy of $\triangle v_1v_2v_3$. We have an interesting observation that if

$$d_1^i > d_1^j, \quad d_2^i > d_2^j, \quad d_3^i > d_3^j,$$

then the i-th copy of $\triangle v_1v_2v_3$ does not contribute to determining the actual ridge curve. Therefore, we advocate extending the surface on the fly. The extension process stops if no triangles can provide geodesic distances that contribute to the actual ridge curve. In other words, there is no need to determine the number of rings to extend the surface in the very beginning.

Remark The selection of geodesic solver primarily depends on the user's varying requirements for accuracy and speed. Using a exact geodesic solver (e.g., MMP algorithm (Mitchell et al., 1987)) yields a more accurate computation of the cut locus. However, this precision comes at the cost of increased computational time. Conversely, approximate geodesic solver (e.g., Fast Marching algorithm (Kimmel and Sethian, 1998)) sacrifice some accuracy but offer faster computation speeds. These approximate solvers are particularly suitable for scenarios prioritizing computational efficiency over absolute precision. By understanding the trade-offs between accuracy and speed, users can select the most suitable geodesic solver for their specific application needs.

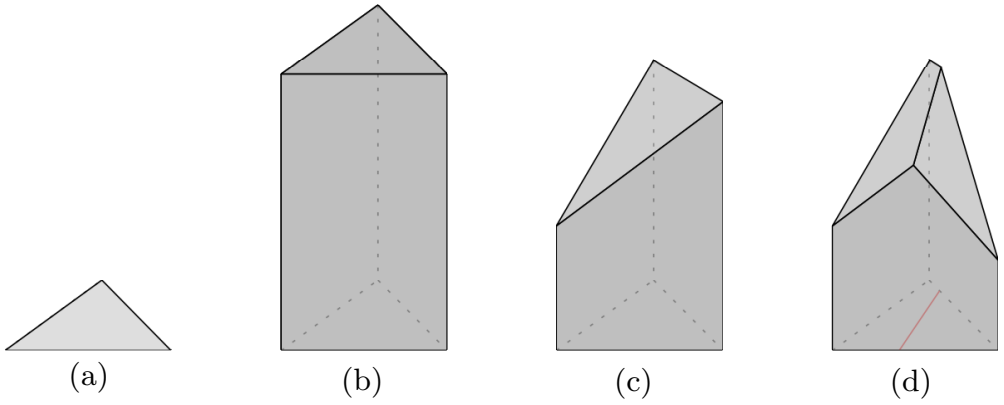


Figure 8: (a) Place a triangle located on the model onto a two-dimensional plane. (b) Using the triangle from (a) as a base, form an infinitely tall triangular prism. (c) Cut the prism using half-planes, preserving the portion below the half-plane. (d) After all the plane cutting operations are done, project the roof structure onto the plane of the triangle, yielding the ridge structure within $\triangle v_1v_2v_3$.

3.2.3. Computing accurate geodesic ridges

To compute the accurate geodesic ridge curve, we adopt the incremental cutting method inspired by Xin et al. (2022). Let d_1^i, d_2^i, d_3^i be the shortest distances at the three vertices of the i-th copy of $\triangle v_1v_2v_3$. We place triangle $\triangle v_1v_2v_3$ on a 2D plane. We can use a half-plane π_i to define the linear field given by d_1^i, d_2^i, d_3^i, where π_i passes through the following three vertices:

$$(x_1, y_1, d_1^i),\ (x_2, y_2, d_2^i),\ (x_3, y_3, d_3^i).$$

Our objective is to find the lower envelope of these half-planes. We construct an infinite triangular prism with $\triangle v_1v_2v_3$ as the base and then incrementally cut the volume by $\{\pi_i\}_{i=1}^K$, following a similar approach to Du et al. (2021). After all the plane cutting operations are done, we project the roof structure onto the plane of the triangle, yielding the ridge structure within $\triangle v_1v_2v_3$. We demonstrate this process in Figure 8.

4. Evaluation

We conducted comparative experiments and demonstrated the utility of our algorithm in several scenarios. The code is written in C++. The results presented below were obtained on a computer equipped with a 2.5GHz Intel i5-12400F CPU running Windows 11 as its operating system.

4.1. Comparison with the state of the arts

Among the existing methods for computing ridges, Mancinelli et al. (2021) stands out due to its impressive computational speed. The method infers ridges by utilizing gradient information within triangles and refines the computed ridges through post-processing to ensure adherence to the correct topology. It provides a practical and computationally efficient tool for extracting ridges. However, due to imprecise gradient information of the computed triangles, the calculated ridges often exhibit significant errors, especially when the triangles are large or the quality of the triangular mesh is poor. Based on our tests, their algorithm is not numerically robust, leading to frequent failures on many models. In Fig. 9, we present a comparison between our algorithm and Mancinelli et al. (2021). It is known that the cusps of geodesic isolines denote non-differential points, indicating the presence of ridges. It can be observed that our algorithm consistently reports the actual ridge curve, while Mancinelli et al. (2021) shows considerable deviation from the real ridges. In Fig. 10, we showcase a gallery of results computed by our algorithm.

4.2. Precise geodesic isolines

Traditional texturing algorithms for visualizing the isolines of a distance field suffer from linear interpolation, resulting in conspicuous artifacts and missing sharp corners. This occurs due to the loss of information within triangles as a result of direct linear interpolation, causing the displayed texture to lose the sharpness of the geodesic isolines.

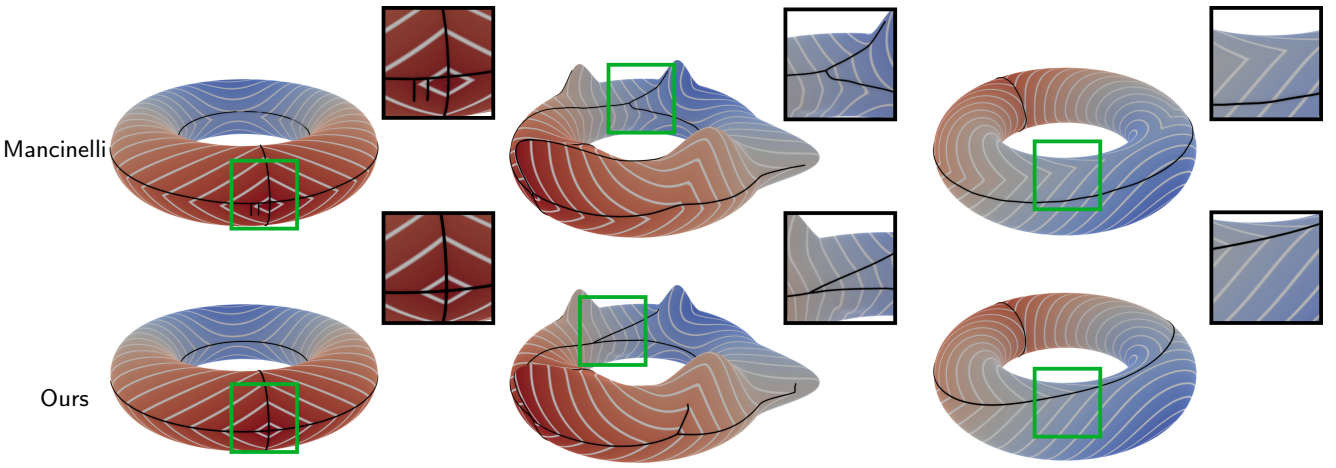


Figure 9: Visual comparison of our method against Mancinelli et al. (2021). Our results are significantly better as they align well with the cusps of the geodesic isolines. In contrast, Mancinelli et al. (2021) produces a conspicuous deviation from the actual setting; See the highlighted windows.

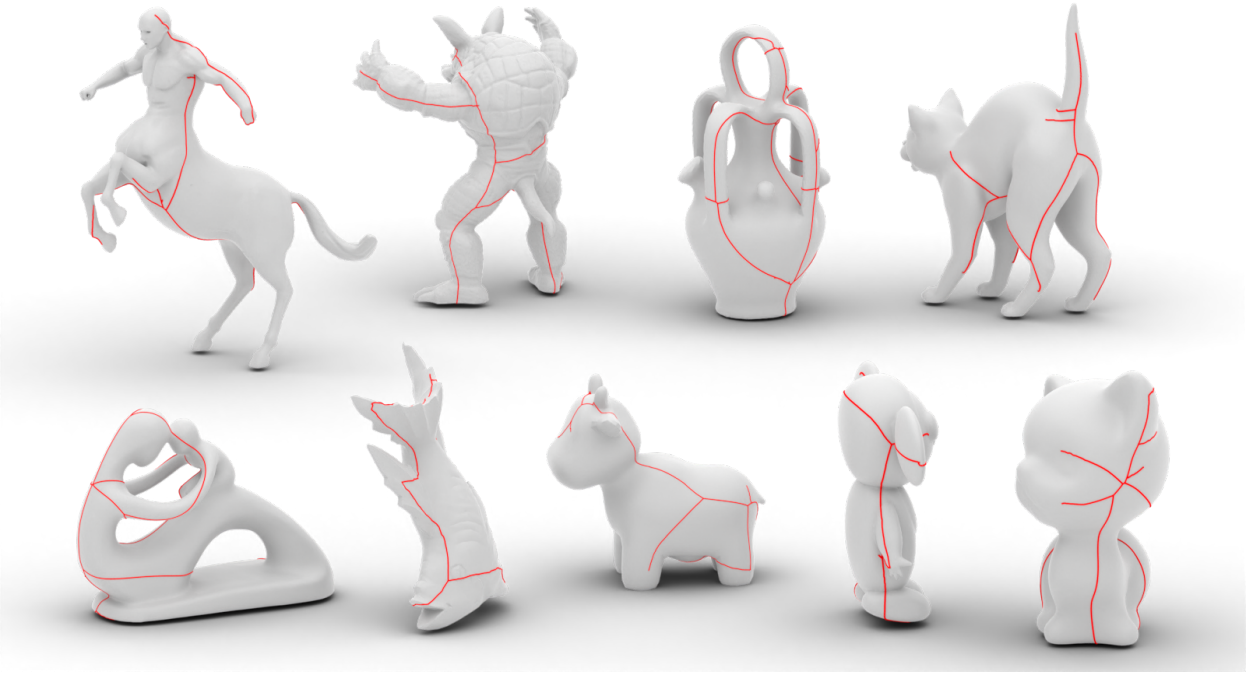


Figure 10: A gallery of ridge results obtained through our algorithm.

However, by embedding the ridge curve into the triangle mesh, the ridges are visually superior; See Figure 11. This contrast validates the idea that the ridge curve is helpful for the piecewise linear representation of the geodesic distance field.

4.3. Precise geodesic distance query

In conventional geodesic distance algorithms, if we need to calculate the distance field within a triangle, the most direct method is to perform linear interpolation using the distance values at the three vertices of the triangle. However, for triangles crossing the ridge curve mentioned above, linear interpolation is not suitable, as the distance change is not

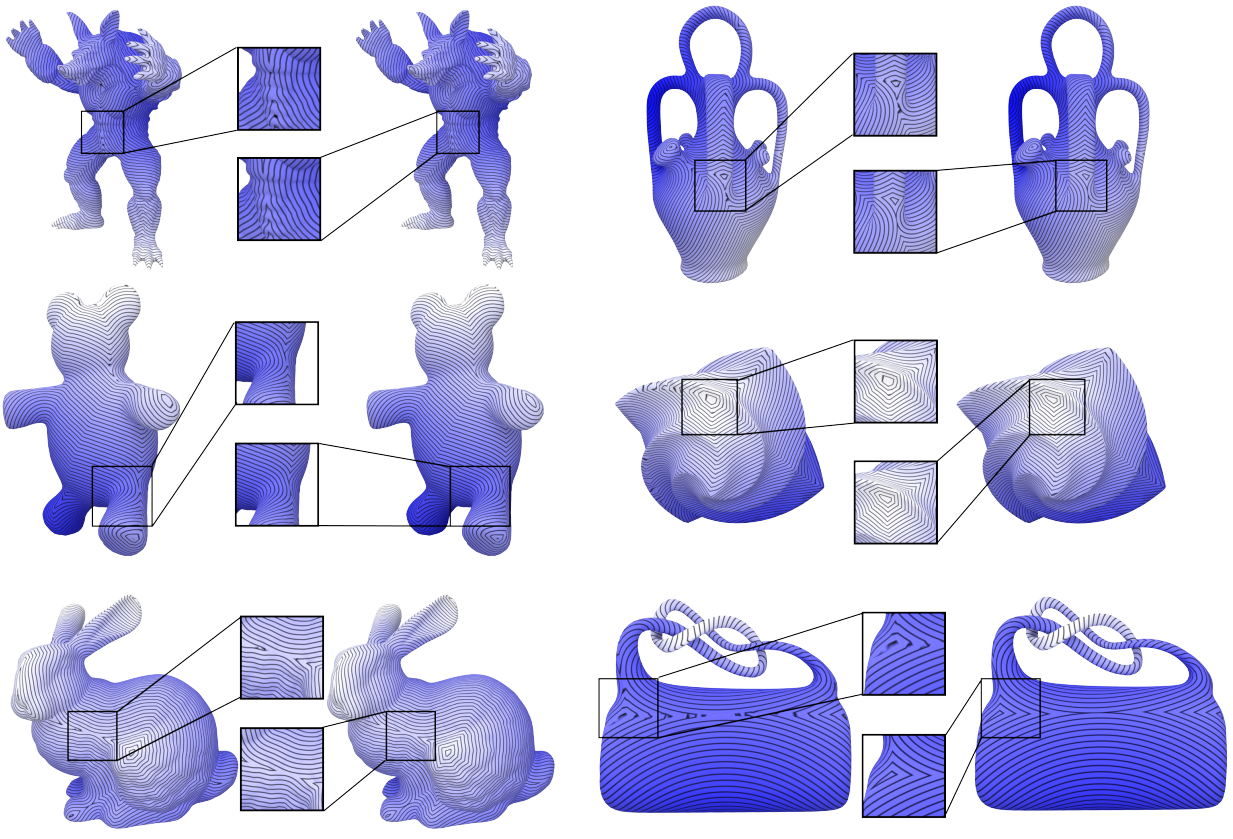


Figure 11: If we directly visualize the distance field using the texturing technique, numerous artifacts are present, as seen in the results on the left. However, by embedding the ridge curve into the triangle mesh, the ridges on the right (with the embedded ridge curve) are visually superior. Note that only triangles crossed by the ridge curve are split into smaller ones, resulting in a slight increase in the number of triangles. This observation validates the idea that the ridge curve is helpful for the piecewise linear representation of the geodesic distance field.

Model		teddy (0.2k)	bottle (1k)	bunny (1k)	cheburashka (3k)	armadillo (4k)	trim_star (10k)	knot_high (16k)
VTP	Mean Error (%)	0.8412	0.9761	0.4427	0.2201	0.1966	0.2243	0.0421
Fast Marching	Mean Error (%)	3.5097	3.8547	2.8011	1.8077	1.9017	1.7450	1.0007
Ours	Mean Error (%)	**0.5037**	**0.7645**	**0.1945**	**0.0884**	**0.0726**	**0.0312**	**0.0150**

Table 1
We embed the sampling points onto the surface and use VTP to obtain accurate distance values for the sampling points, serving as the benchmark for comparison. Subsequently, we calculate the average error of the distance values for each algorithm by evaluating all sampling points. The highest precision scores are highlighted in bold.

linear in these triangles. Linear interpolation cannot approximate the real distance change, leading to low estimation precision.

To better demonstrate the advantages of our algorithm, we randomly select some points on the surface of the original model and then embed them into the mesh surface, making the sampled points become vertices. By this trick, we can get the ground-truth distance values for comparison. We can see from Table 1 that even if we run the VTP algorithm to accurately compute the distance to mesh vertices, linear interpolation still causes low precision. In contrast, if we further slice each triangle along the ridge curve, the query accuracy significantly increases; See the highest precision scores highlighted in bold in Table 1.

4.4. Ablation study

Results under different tolerance. In the preceding sections, we defined the significance $I(uv)$ of edge $uv \in T^*$ and explained how we can prune the dual structure by setting a tolerance, allowing for a clearer and simpler presentation

of the rough ridge lines obtained in the first step. In the specific implementation, we introduced a threshold for the ratio of the area represented by significance, denoted as $I(uv)$, to the surface area Area_{total} of the model. For parts where the ratio $I(uv)/\text{Area}_{total}$ is less than the given threshold tol, we directly discard them. The tolerance holds significant importance in our approach. As the tolerance value decreases, the computed cut locus covers more regions of the model. An example is presented in Fig. 12, demonstrating the variations in the computed cut locus as the tolerance decreases. It can be observed that reasonable results are obtained at any tolerance. For practical applications, we recommend setting the default tolerance value to 0.025, which yields reasonable results in the vast majority of scenarios.

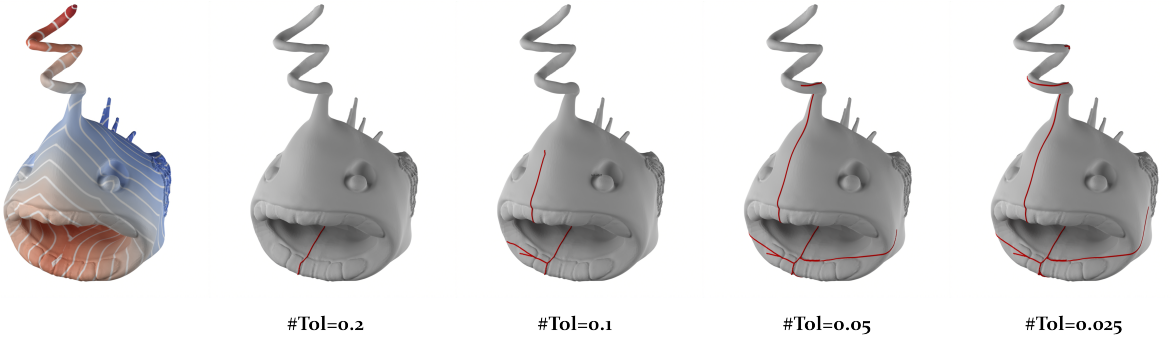


Figure 12: In the figure, we demonstrate the impact of setting different tolerances on the resulting exact ridge lines. Here, we set a threshold for the ratio of significance I to the total model surface area to ensure that the same threshold is applicable to most models. The threshold on the far left is set too small, resulting in a large number of ridge lines being trimmed, while the threshold on the far right is set too large, causing some less significant ridge lines to be retained. However, our definition of significance ensures that the resulting limit structures are always continuous.

Cut locus for high complex models. To further validate our algorithm, we applied it to compute the cut locus of various complex models, including geometrically intricate structures and high-genus models, among others, as shown in Fig. 13. This experimentation aims to demonstrate the algorithm's efficacy across a diverse range of challenging scenarios.

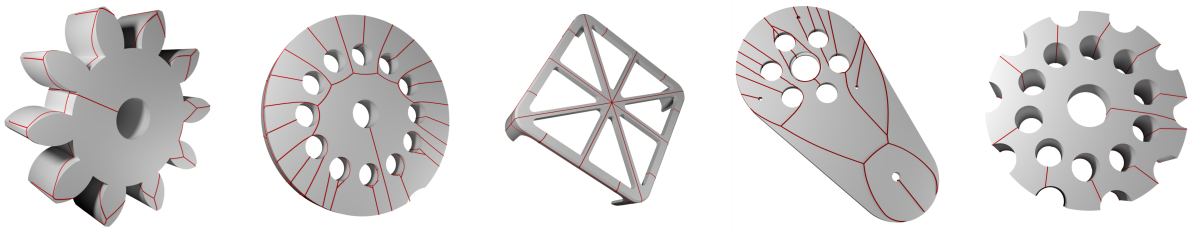


Figure 13: Presentation of cut locus results for complex models.

Results under different mesh resolutions. In the algorithmic workflow, triangles are assumed to be the fundamental units where the distance field undergoes linear changes. Therefore, the resolution of the model significantly influences the computation results. By adjusting the resolution of the model, we computed the cut locus results separately, aiming to investigate the impact of model resolution on cut locus results. The results are illustrated in Fig. 14.

5. Conclusion, limitations and future work

In this paper, we propose a numerically robust method for computing geodesic ridge curves, which is a notoriously hard problem. With the help of extracted ridge curves, we can yield a region-wise linear representation of the geodesic distance field. Our algorithm consists of two steps: first finding a rough ridge curve and then refining it into the correct configuration by simply calling an accurate geodesic solver.

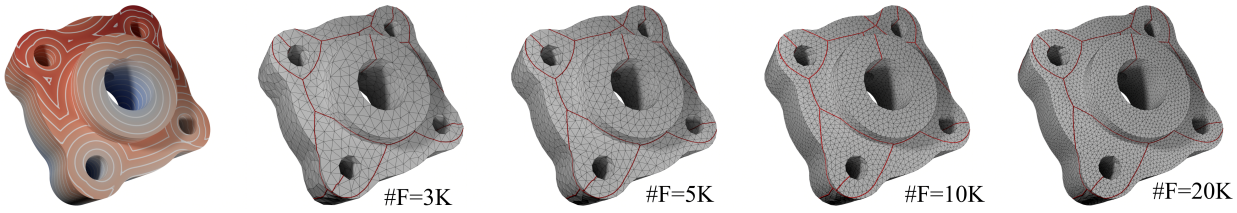


Figure 14: Presentation of cut locus results obtained at different model resolutions. We annotated the number of Faces (F) in the figure.

Our algorithm possesses at least two key properties that distinguish it from existing algorithms. First, we introduce a pruning mechanism such that an arbitrary pruning parameter can ensure the connectivity of the ridge curve. Second, our algorithm supports any geodesic solver to provide geodesic distances. Even with the fast marching method, our algorithm can yield a smooth and clean ridge curve that aligns well with the cusps of geodesic isolines. We demonstrate its utility in accurate geodesic distance querying and high-fidelity visualization of geodesic iso-lines.

Our algorithm, still needs to be improved. Firstly, our algorithm is relatively slow and can be further accelerated. Secondly, our current algorithm is designed for triangle mesh, with limited scalability. In the future, we will explore the possibility of speeding up the computation and extending our algorithm to parameter surfaces.

References

Buchner, M.A., 1977. Simplicial structure of the real analytic cut locus. Proceedings of the American Mathematical Society 64, 118–121.

Chen, J., Han, Y., 1990. Shortest paths on a polyhedron, in: Proceedings of the sixth annual symposium on Computational geometry, pp. 360–369.

Crane, K., Weischedel, C., Wardetzky, M., 2013. Geodesics in heat: A new approach to computing distance based on heat flow. ACM Transactions on Graphics (TOG) 32, 1–11.

Du, X., Zhou, Q., Carr, N., Ju, T., 2021. Boundary-sampled halfspaces: a new representation for constructive solid modeling. ACM Transactions on Graphics (TOG) 40, 1–15.

Erickson, J., Har-Peled, S., 2002. Optimally cutting a surface into a disk, in: Proceedings of the eighteenth annual symposium on Computational geometry, pp. 244–253.

Erickson, J., Whittlesey, K., 2005. Greedy optimal homotopy and homology generators, in: SODA, pp. 1038–1046.

Générau, F., Oudet, E., Velichkov, B., 2022. Cut locus on compact manifolds and uniform semiconcavity estimates for a variational inequality. Archive for Rational Mechanics and Analysis 246, 561–602.

Itoh, J.i., Sinclair, R., 2004. Thaw: A tool for approximating cut loci on a triangulation of a surface. Experimental Mathematics 13, 309–325.

Kimmel, R., Sethian, J.A., 1998. Computing geodesic paths on manifolds. Proceedings of the national academy of Sciences 95, 8431–8435.

Kutz, M., 2006. Computing shortest non-trivial cycles on orientable surfaces of bounded genus in almost linear time, in: Proceedings of the 22nd ACM Symposium on Computational Geometry, Sedona, Arizona, USA, June 5-7, 2006.

Liu, Y.J., 2013. Exact geodesic metric in 2-manifold triangle meshes using edge-based data structures. Computer-Aided Design 45, 695–704.

Mancinelli, C., Livesu, M., Puppo, E., 2021. Practical computation of the cut locus on discrete surfaces, in: Computer Graphics Forum, Wiley Online Library. pp. 261–273.

Misztal, M.K., Bærentzen, J.A., Anton, F., Markvorsen, S., 2011. Cut locus construction using deformable simplicial complexes, in: 2011 Eighth International Symposium on Voronoi Diagrams in Science and Engineering, IEEE. pp. 134–141.

Mitchell, J.S., Mount, D.M., Papadimitriou, C.H., 1987. The discrete geodesic problem. SIAM Journal on Computing 16, 647–668.

Myers, S.B., 1935. Connections between differential geometry and topology. i. simply connected surfaces .

Poincaré, H., 1905. Sur les lignes géodésiques des surfaces convexes. Transactions of the American Mathematical Society 6, 237–274.

Qin, Y., Han, X., Yu, H., Yu, Y., Zhang, J., 2016. Fast and exact discrete geodesic computation based on triangle-oriented wavefront propagation. ACM Transactions on Graphics (TOG) 35, 1–13.

Rote, M.G., 2003. Shortening of curves and decomposition of surfaces. Ph.D. thesis. Citeseer.

Sakai, T., 1996. Riemannian geometry. volume 149. American Mathematical Soc.

Sinclair, R., Tanaka, M., 2002. Loki: Software for computing cut loci. Experimental Mathematics 11, 1–25.

Surazhsky, V., Surazhsky, T., Kirsanov, D., Gortler, S.J., Hoppe, H., 2005. Fast exact and approximate geodesics on meshes. ACM transactions on graphics (TOG) 24, 553–560.

Tao, J., Zhang, J., Deng, B., Fang, Z., Peng, Y., He, Y., 2019. Parallel and scalable heat methods for geodesic distance computation. IEEE transactions on pattern analysis and machine intelligence 43, 579–594.

Thomassen, C., 1990. Embeddings of graphs with no short noncontractible cycles. Journal of Combinatorial Theory, Series B 48, 155–177.

Xin, S., Wang, P., Xu, R., Yan, D., Chen, S., Wang, W., Zhang, C., Tu, C., 2022. Surfacevoronoi: Efficiently computing voronoi diagrams over mesh surfaces with arbitrary distance solvers. ACM Transactions on Graphics (TOG) 41, 1–12.

Xin, S.Q., Wang, G.J., 2009. Improving chen and han's algorithm on the discrete geodesic problem. ACM Transactions on Graphics (TOG) 28, 104.

Xu, C., Wang, T.Y., Liu, Y.J., Liu, L., He, Y., 2015. Fast wavefront propagation (fwp) for computing exact geodesic distances on meshes. IEEE transactions on visualization and computer graphics 21, 822–834.

A. Pseudocode

Algorithm 1: Area Accumulation Scheme

Input: Dual structure T^*, Areas for each face A
Output: Accumulation values Acc at each vertex of T^*

$V_1 \leftarrow$ The degree 1 nodes in T^*;
$\text{Area}_{total} \leftarrow \sum A(f)$;
for $v \in V_1$ **do**
    $\text{Acc}[v] = \text{A}[v]$;
    $v_{last} \leftarrow v$;
    $v_{next} \leftarrow$ next vertex for v;
    while $Degree(v_{next}) = 2$ **do**
        $\phi[v_{next}] \leftarrow \text{Acc}[v_{last}] + \text{A}[v_{next}]$;
        $\text{Acc}[v_{next}] \leftarrow \min(\phi[v_{next}],\ \text{Area}_{total} - \phi[v_{next}])$;
        $v_{last} \leftarrow v_{next}$;
        $v_{next} \leftarrow$ next vertex for v_{next};
    end
    if $Degree(v_{next}) = 3$ **then**
        if v_{next} *has a recorded values* $\phi[v]$ *in the map* **then**
            $\phi[v_{next}] \leftarrow \phi[v] + \phi[v_{last}] + \text{A}[v_{next}]$;
            $\text{Acc}[v_{next}] \leftarrow \min(\phi[v_{next}],\ \text{Area}_{total} - \phi[v_{next}])$;
            $v_{last} \leftarrow v_{next}$;
            $v_{next} \leftarrow$ unvisited neighbor;
            Goto 7;
        else
            store $\phi[v_{last}]$ in the map with v_{next} as the key;
        end
    end
end