Computational Visual Media
https://doi.org/10.1007/s41095-022-0326-0 Vol. 9, No. 3, September 2023, 443–459

Research Article

An efficient algorithm for approximate Voronoi diagram construction on triangulated surfaces

Wenlong Meng[1], Pengbo Bo[1], Xiaodong Zhang[1], Jixiang Hong[2], Shiqing Xin[2] (✉), and Changhe Tu[2] (✉)

Abstract Voronoi diagrams on triangulated surfaces based on the geodesic metric play a key role in many applications of computer graphics. Previous methods of constructing such Voronoi diagrams generally depended on having an exact geodesic metric. However, exact geodesic computation is time-consuming and has high memory usage, limiting wider application of geodesic Voronoi diagrams (GVDs). In order to overcome this issue, instead of using exact methods, we reformulate a graph method based on Steiner point insertion, as an effective way to obtain geodesic distances. Further, since a bisector comprises hyperbolic and line segments, we utilize Apollonius diagrams to encode complicated structures, enabling Voronoi diagrams to encode a medial-axis surface for a dense set of boundary samples. Based on these strategies, we present an approximation algorithm for efficient Voronoi diagram construction on triangulated surfaces. We also suggest a measure for evaluating similarity of our results to the exact GVD. Although our GVD results are constructed using approximate geodesic distances, we can get GVD results similar to exact results by inserting Steiner points on triangle edges. Experimental results on many 3D models indicate the improved speed and memory requirements compared to previous leading methods.

Keywords geodesic Voronoi diagrams (GVDs); triangular surfaces; mesh surfaces; approximate geodesics; Apollonius diagrams

1 School of Computer Science and Technology, Harbin Institute of Technology, Weihai 264209, China. E-mail: W. Meng, mengwenlong_hit@163.com; P. Bo, bob_pengbo@163.com; X. Zhang, zxd@hit.edu.cn.

2 School of Computer Science and Technology, Shandong University, Qingdao 266237, China. E-mail: J. Hong, jxhong1230@gmail.com; S. Xin, xinshiqing@sdu.edu.cn (✉); C. Tu, chtu@sdu.edu.cn (✉).

Manuscript received: 2022-09-16; accepted: 2022-11-21

1 Introduction

Voronoi diagrams are fundamental data structures that have been widely investigated in computational geometry, and applied in various science and engineering disciplines, e.g., molecular biology [1], image processing [2, 3], computer graphics [4–6], machining and manufacturing [7], and mobile communication [8], etc. A Voronoi diagram can be viewed as the minimization diagram of a finite ensemble of continuous functions [9]. A Voronoi diagram divides the embedding space into sub-regions, each being the domain closer to a given seed point than the others. We may define many variants of Voronoi diagrams depending on the embedding space. In Euclidean space, the Voronoi diagram is well understood and studied for rasterization data [10, 11]. For spaces with non-Euclidean metrics, people have developed Voronoi diagrams on parametric surfaces [12], polyhedral surfaces [13, 14], spherical spaces [15], hyperbolic spaces [16], etc. However, there are distinct differences between Euclidean and non-Euclidean metrics, so many established attributes based on the Euclidean metric do not hold for non-Euclidean metrics. For instance, a 2D Euclidean Voronoi component is always convex, while a geodesic Voronoi component is often non-convex. In this paper, we specifically consider Voronoi diagrams embedded in triangular meshes, based on the geodesic metric (geodesic Voronoi diagrams, GVDs).

Owing to its simplicity and flexibility, the triangular mesh is the most frequently used 3D representation in computer graphics [17]. Voronoi diagrams on triangular meshes have been used in a range of applications, such as 3D mesh reconstruction [18, 19], tree skeleton extraction [13], trajectory planning [20],

shape segmentation [21], and so on. However, making use of geodesic Voronoi diagrams on triangular surfaces, rather than diagrams based on the Euclidean metric, presents many unresolved challenges. Often, in order to construct geodesic Voronoi diagrams, exact geodesic algorithms are used, such as the MMP algorithm [22] and the VTP algorithm [23], to obtain the geodesic metric [13, 14, 24]. Due to the excellent features of these algorithms, they can supply adequate geodesic information about edges for GVD construction. However, since these exact algorithms are slow and memory hungry, they present a stumbling block to constructing GVDs and further practical applications. Furthermore, the boundaries of GVD regions, bisectors, consist of hyperbolic and line segments that are have particular, more complex characteristics than those of Euclidean Voronoi diagrams.

In this paper, we focus on avoiding superfluous computation of geodesic distances to improve speed and memory usage. As computing geodesic distances using exact algorithms is expensive, we construct graphs using Steiner points placed on mesh edges to report distances. In the propagation process, we regard each Steiner point as a simplified window and adapt existing window filtering rules to remove many redundant events. Theoretically, a bisector in a Voronoi diagram on a mesh may include linear, hyperbolic, and parabolic segments. We make use of the Apollonius diagram, also called the additively weighted Voronoi diagram, to depict these structures locally defined in the plane of a single mesh triangle.

Based on these ideas, we present an efficient method for constructing an approximate GVD for point-source generators. Once distance propagation terminates, geodesic distances and the GVD become concurrently available. Thanks to the highly selective filtering rules, our algorithm runs in $O(mn\log n)$ time and takes $O(mn)$ space on an n-face mesh with t generators, where m is the number of Steiner points inserted on each edge. Figure 1 is a representative of experimental results on a large number of 3D models, indicating that our algorithm has improved speed and memory efficiency compared over previous leading methods.

In summary, our key ideas include the following:

(1) Unlike previous GVD construction methods, we reformulate a Steiner point insertion method, instead of using exact methods, to obtain geodesic distances, increasing speed and reducing memory usage.

(2) We utilize local Apollonius diagrams with weighted Steiner points to partition mesh triangles and extract hyperbolic and linear segments, to encode the complicated GVD bisector structures.

(3) We suggest a method of approximate GVD evaluation based on discrete Fréchet distance, and use it to show that our approximate GVD results are close to the exact results.

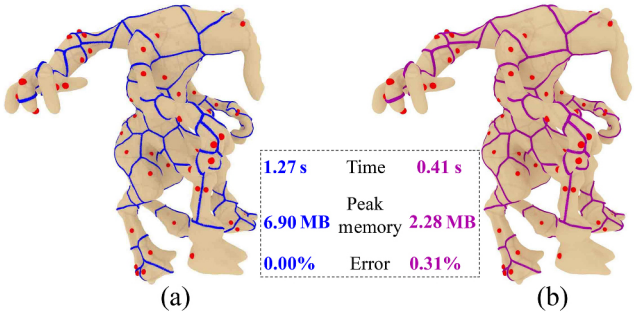


Fig. 1 GVD computed by (a) Qin et al.'s method and (b) our method, for the Monster model (50k faces), for 30 generators. Qin et al.'s method takes 1.27 s and 6.90 MB memory to get an exact GVD; ours takes 0.41 s and 2.28 MB memory to construct a GVD with 0.31% error.

Our work is presented in detail as follows. Section 2 briefly recalls related work. Section 3 provides background on the Apollonius diagram and geodesic distance computation based on Steiner points. Section 4 explains how to construct a GVD using point-source sites. We explain our algorithm for building GVD on triangle meshes in detail in Section 5. Section 6 experimentally compares our method to previous methods. We give some applications of the proposed GVD algorithm in Section 7.

2 Related work

2.1 Discrete geodesics

The discrete geodesic problem (DGP) focuses on computing geodesic distances on triangle meshes [22]. There are three main kinds of methods for solving it: discrete wavefront propagation methods, PDE methods, and graph-based methods. We refer to Refs. [25, 26] for a comprehensive survey. Discrete wavefront propagation methods use a window to encode geodesic paths sharing the same edge sequence. They generally propagate wavefronts across

triangle faces in a Dijkstra-like sweep. The MMP algorithm [22, 27] and the CH algorithm [28] are two representative methods. The MMP algorithm maintains a partitioning scheme for each directed edge while the CH algorithm uses a *one angle one split* rule so that at most $O(n)$ windows are allowed to have two children. Many variants and improvements [23, 29–32] propose different rules to filter redundant windows in the propagation process. Although discrete wavefront propagation methods can provide exact geodesic information and adequate information for bisectors, these methods are computationally expensive.

PDE methods [33–36] compute geodesic distances using partial differential equations (PDEs) on a smooth manifold. In detail, they solve the Eikonal equation $\|\nabla u(t)\| = 1$ with boundary condition $u(s) = 0$ on discrete domains, where s is the source. PDE methods are easy to implement and efficient. However, they provide only a first-order approximation, which is extremely sensitive to the mesh tessellation.

Graph-based algorithms [37–44] are approximation algorithms; they can balance accuracy and speed very well. These algorithms are characterized by an approximation ratio ϵ. A typical goal is to compute a $(1+\epsilon)$-approximate geodesic path as quickly as possible. Graph-based algorithms usually deal with the discrete geodesic problem by constructing a dense undirected graph G on the target manifold mesh and use graph-based search techniques to answer the distance-and-path query. In general, approaches add Steiner points on mesh edges or even on triangle faces. For example, Lanthier et al. [39, 40] constructed a dense graph by inserting Steiner points on each mesh edge to achieve accurate results. Aleksandrov et al. [41, 45] proposed adding Steiner points more densely near vertices than near mid-edges to ensure that the length of the approximate path is within a factor $(1+\epsilon)$ of the shortest geodesic path. Later, they [42] found that adding Steiner points along the bisectors of triangles also works. A survey on graph-based methods is presented in Ref. [25]. Generally, the Steiner-point insertion algorithms need to construct a graph G in which the complexity of G grows quadratically with respect to the number m of Steiner points inserted on each mesh edge. Consequently, the computational cost grows rapidly as m increases. Since efficiently computing geodesic is crucial in GVD computation, we reformulate a graph method [46] based on Steiner point insertion; we regard each Steiner point as a simplified window and use existing window filtering rules for efficiently propagating geodesic distances.

2.2 Voronoi diagram

Mathematically, a Voronoi diagram is a partition of the plane into regions, each closer to one of a given set of objects. Using the same definition, the construction of Voronoi diagrams can be extended to metric spaces other than Euclidean spaces [47, 48]. For non-Euclidean spaces, there are many works dedicated to Voronoi diagrams on smooth surfaces, e.g., Riemannian manifolds $\mathbb{M}$ [49, 50], parametric surfaces [12], spheres $\mathbb{S}^2$ [15, 51], and hyperbolic spaces $\mathbb{H}^2$ [16]. The reader is referred to Refs. [10, 52] for detailed surveys.

Owing to their simplicity and a high degree of freedom, non-differentiable polyhedral surfaces, especially triangular surfaces, are the most popular representation for 3D objects. Therefore, constructing a Voronoi diagram on triangle mesh surfaces using the geodesic metric is an important topic in computer graphics. Since many properties of a smooth manifold no longer hold, transferring Voronoi diagrams from Euclidean metric to geodesic metric still presents many unresolved challenges. Kimmel and Sethian [53] computed Voronoi diagrams on triangle meshes using the fast marching method [33]. However, since that method computes only a first-order approximation to geodesic distance, it may produce bad results on poor triangulations. Liu et al. [13] studied the analytic structure of iso-contours, bisectors, and Voronoi diagrams on a triangular mesh M. They proposed practical algorithms for constructing GVDs using the MMP algorithm for exact geodesic computation. Later, they extended their work to construct intrinsic Delaunay triangulations from dual geodesic Voronoi diagrams [54, 55]. Xu et al. [24] proposed an algorithm for constructing GVDs with polyline generators. They introduced a new concept, the local Voronoi diagram (LVD), to represent the GVD bisectors. Recently, Qin et al. [14] proposed constructing Voronoi diagrams using the window-VTP algorithm which runs 3–8 times faster than Liu et al.'s

method [13]. Although exact geodesic algorithms like MMP and VTP can compute Voronoi diagrams accurately, obtaining exact geodesic distances is time-consuming, presenting a stumbling block to constructing GVDs and further practical applications. In this paper, we employ an improved Steiner-point graph method to balance time and accuracy of GVD construction.

3 Preliminaries

3.1 Steiner points

In computational geometry, a Steiner point is a point that is not part of the input to a geometric problem but is added during the solution of the problem. It usually creates a better solution than using the original points alone [56]. In geodesic computation, Steiner points are regularly applied in graph-based methods [39, 40, 42] to balance accuracy and time. By inserting m Steiner points into each mesh edge, and connecting any pair of Steiner points in the same triangle, we can construct an augmented graph G. In the original Steiner-point graph algorithms [39–42], when a Steiner point p is assigned a shorter distance, it must broadcast an update event to all of the neighboring Steiner points and vertices: see Fig. 2. This causes a rapid rise in the computational cost since the complexity of G is $O(m^2)$ where m is the number of Steiner points inserted per edge.

In order to seek a balance between accuracy and speed in such graph algorithms, Meng et al. [46] proposed an improved Steiner-point graph method. In it, the inserted Steiner points play much like the same role as windows in exact geodesic methods. We can filter useless broadcast events by adapting existing window pruning rules. In order to efficiently construct Voronoi diagrams based on the geodesic metric, we adopt filtering strategies and revise Meng et al.'s original method for computation of the multi-source geodesic distance field.

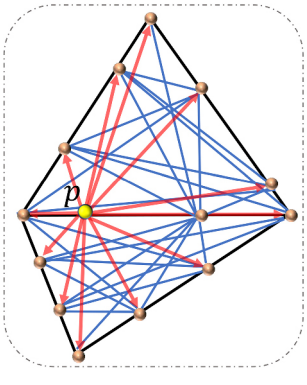


Fig. 2 Steiner points and propagation.

3.2 Apollonius diagram

The Voronoi diagram is a fundamental data structure defined as the minimization diagram of a finite set of continuous functions [10, 52, 57]. It subdivides the embedding space $\Omega \subset \mathbb{R}^d$ into certain regions based on the Euclidean distance to a set of points $\{x_i\}_{i=1}^t$ belonging to Ω. Each point $\{x_i\}_{i=1}^t$, called a site or generator, dominates the subregion:

$$VC(x_i) = \{x \in \Omega \mid \| x - x_i \| \leqslant \| x - x_j \|,\ i \neq j\} \quad (1)$$

This is called the Voronoi cell of the site x_i. We may define many variants of Voronoi diagrams using different distance functions and embedding spaces. Power diagrams and Apollonius diagrams are two classical variants of Voronoi diagrams (see Fig. 3).

A power diagram is a type of weighted Voronoi diagram. Instead of each region comprising the points closest to a site, it comprises those points with smallest power distance for a particular circle [58, 59]. Each site x_i with weight w_i dominates the subregion or power cell:

$$PC(x_i) = \{x \in \Omega \mid \| x - x_i \|^2 - w_i \leqslant \| x - x_j \|^2 - w_j,\ i \neq j\} \quad (2)$$

A power diagram degenerates to a Voronoi diagram when the weights are identical.

The Apollonius diagram, or additively weighted Voronoi diagram, is another variant [60–62]. Differing from the power diagram which uses squared distance, each site x_i with weight w_i dominates the subregion, or Apollonius cell:

$$AC(x_i) = \{x \in \Omega \mid \| x - x_i \| - w_i \leqslant \| x - x_j \| - w_j,\ i \neq j\} \quad (3)$$

The sets of points simultaneously belonging to two Apollonius cells are called Apollonius edges. The

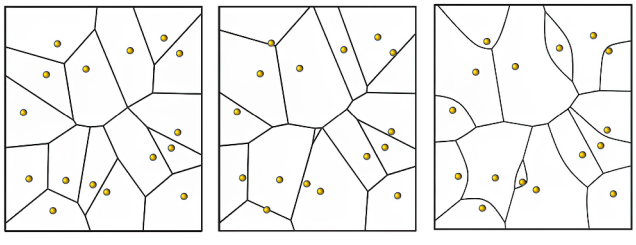


Fig. 3 Left to right: 2D Voronoi diagram, power diagram, and Apollonius diagram for the same set of sites.

Apollonius edges between two adjacent cells usually contain linear and hyperbolic segments, and is the motivation for applying Apollonius diagrams rather than other Voronoi diagram variants to construct the GVD.

4 GVD with point-source sites

Given a set of sites $S = \{s_1, \cdots, s_t\}$ on mesh $\mathcal{M}$, let $G_{s_i}(p)$ denote the geodesic distance from site s_i to point p on $\mathcal{M}$. The geodesic Voronoi cell associated with site s_i can be defined as

$$GC(s_i) = \{p \mid G_{s_i}(p) \leqslant G_{s_j}(p),\ i \neq j,\ p \in \mathcal{M}\} \quad (4)$$

Furthermore [14], the boundaries of geodesic Voronoi cells are given by the sets of points q satisfying:

$$\exists\, i, j,\ \forall k:\ G_{s_i}(q) = G_{s_j}(q) \leqslant G_{s_k}(q),\ i \neq j \neq k \quad (5)$$

The common boundary of sites s_i and s_j can also be regarded as their bisector.

4.1 Bisectors

A bisector $\mathcal{B}(s_i, s_j)$ of two sites $s_i, s_j \in S$ is the locus of points q on mesh $\mathcal{M}$ satisfying $G_{s_i}(q) = G_{s_j}(q)$. Property 4.1 states that boundaries, i.e., bisectors, in GVDs, consist of hyperbolic and line segments. These are distinctive and more complicated than those of Euclidean Voronoi diagrams.

Property 4.1 *Given a triangle mesh $\mathcal{M} = (V, E)$ and distinct sites s_1, s_2, let p be a point on the bisector $\mathcal{B}(s_1, s_2)$. Let each Steiner point q_i be an intermediate point on some path from site S_i $\gamma(s_i, p)$, $i = 1, 2$. We assume each point q_i has a known geodesic distance $G_{s_i}(q_i)$ from site s_i. The bisector $\mathcal{B}(s_1, s_2)$ on $\mathcal{M}$ consists of hyperbolic and line segments.*

Proof. The key is to analyze the bisectors as they appear in a triangle $\mathcal{F}$. In the Steiner-point insertion method, $m \geqslant 0$, Steiner points are inserted into each triangle edge. There are two cases to consider according to whether the Steiner points used as relays to compute distance to p have equal or unequal distances to their respective sites:

- Case 1: when $G_{s_1}(q_1) = G_{s_2}(q_2)$, point p lies on the line segment bisecting sites s_1 and s_2.
- Case 2: when $G_{s_1}(q_1) \neq G_{s_2}(q_2)$, point p satisfies $G_{s_1}(q_1) + \|q_1 p\| = G_{s_2}(q_2) + \|q_2 p\|$, which implies that p is on a hyperbolic segment with foci q_1 and q_2.

If three or more Steiner points give the same distance to point p, it is a branch point. This completes the proof. Figure 4 illustrates the above situations. □

In order to find the boundaries of GVDs, we employ Apollonius diagrams to guide the capture of these complicated structures, as we next explain.

4.2 Local Apollonius diagrams

Unlike traditional Voronoi diagrams, Apollonius diagrams consist of linear and hyperbolic segments. Let $p \in \mathcal{B}(s_1, s_2)$ be a point located in triangle $\mathcal{F}$ while the Steiner point q_i, inserted on an edge of triangle $\mathcal{F}$, is a relay point on path $\gamma(s_i, p)$, $i = 1, 2$. Let q_i be provided with a geodesic distance $G_{s_i}(q_i)$ from site s_i to q_i. We have

$$G_{s_1}(q_1) + \|q_1 p\| = G_{s_2}(q_2) + \|q_2 p\| \quad (6)$$

Taking $(q_1, -G_{s_1}(q_1))$ and $(q_2, -G_{s_2}(q_2))$ as two weighted points, their additively weighted distances to p are respectively

$$\|q_1 p\| - (-G_{s_1}(q_1)), \quad \|q_2 p\| - (-G_{s_2}(q_2))$$

Thus, point p has an equal-weighted distance to sites s_1 and s_2 via Steiner points q_1 and q_2, respectively. More generally, bisectors $\mathcal{B}$ (may be with a tree-like structure) can be reported by computing the Apollonius diagram with respect to a set of Steiner points on the boundary of triangle $\mathcal{F}$.

We may represent a triangle by its vertices: $\mathcal{F} = (v_1, v_2, v_3)$. Let $\mathcal{Q} = \{q_k \mid \forall k,\ q_k \in e(v_i, v_j),\ i, j = 1, 2, 3, i \neq j\}$ represent the set of Steiner points inserted into the edges of $\mathcal{F}$. When the Steiner-point graph method terminates, each Steiner point has a geodesic distance $G_{s_i}(q_k)$ measured from generator s_i, $i = 1, \cdots, t$. Taking the set of weighted points $\{q_k, -G_{s_i}(q_k)\}_{k=1}^{M}$ to compute the local Apollonius diagram on $\mathcal{F}$, we can partition $\mathcal{F}$ into M regions, where M cannot exceed the total number of inserted

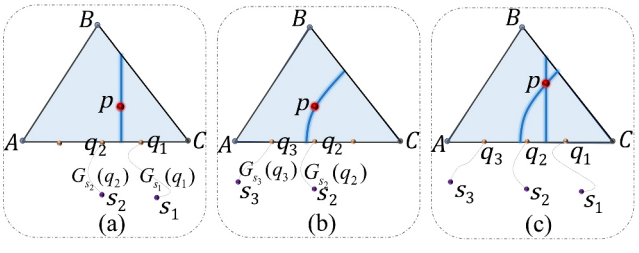


Fig. 4 Bisectors of generators. (a) When $G_{s_1}(q_1) = G_{s_2}(q_1)$, point p is on the line segment bisecting sites s_1 and s_2. (b) When $G_{s_2}(q_2) \neq G_{s_3}(q_3)$, point p is on a hyperbolic segment with foci q_2 and q_3. (c) Point p is a branch point when three or more Steiner points have the same distance to it.

Steiner points. The boundaries of regions are enclosed by line segments and hyperbolic segments. Each segment lies on the border of two distinct regions belonging to different generators, and these form the boundaries of the GVD. See Fig. 5 for an example. Therefore, using the strategy above, by constructing the local Apollonius diagram we can extract the GVD structure triangle by triangle on mesh $\mathcal{M}$.

4.3 Redundant primitives

While this approach to constructing the geodesic Voronoi diagram on a mesh $\mathcal{M}$ could be used in a direct way by first computing the Apollonius diagram on each mesh face, and then finding diagram edges that belong to the GVD, this is time-consuming, since typically only a handful of mesh triangles include GVD boundaries. To develop an efficient method to construct the GVD of mesh $\mathcal{M}$, it is critical to distinguish those triangles which include the GVD boundaries [14], instead of conducting a brute force search over all triangles.

Figure 6 shows a point p which is the intersection between a triangle edge and a GVD boundary. Point p must satisfy the constraint in Eq. (5) and is shared by two adjacent Steiner points connected to two different sources. We call points like p *critical* points; triangles including GVD boundaries always contain such points. In other words, any triangle incorporating GVD structures has Steiner points propagated from different sources, giving criteria for determining the redundant primitives on a mesh:

Given a triangle mesh $\mathcal{M} = (V, E)$ and distinct

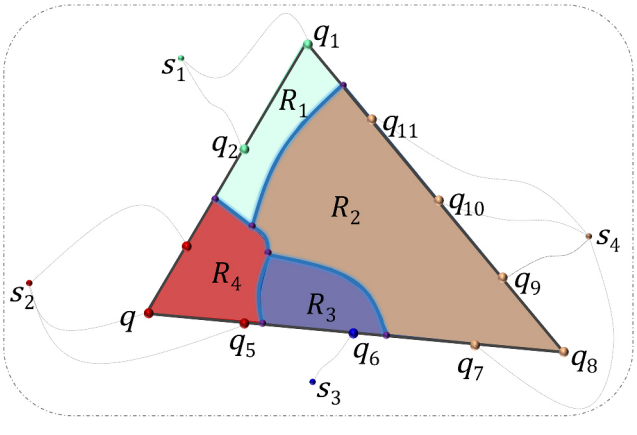


Fig. 5 Local Apollonius diagram on a triangle $\mathcal{F}$. $\mathcal{Q} = \{q_1, \ldots, q_{11}\}$ represents the set of Steiner points inserted into the edges of $\mathcal{F}$ while a set of sites $S = \{s_1, \cdots, s_t\}$ are the generators. The local Apollonius diagram partitions $\mathcal{F}$ into M regions. The boundaries of regions enclosed by line and hyperbolic segments are a part of the mesh GVD.

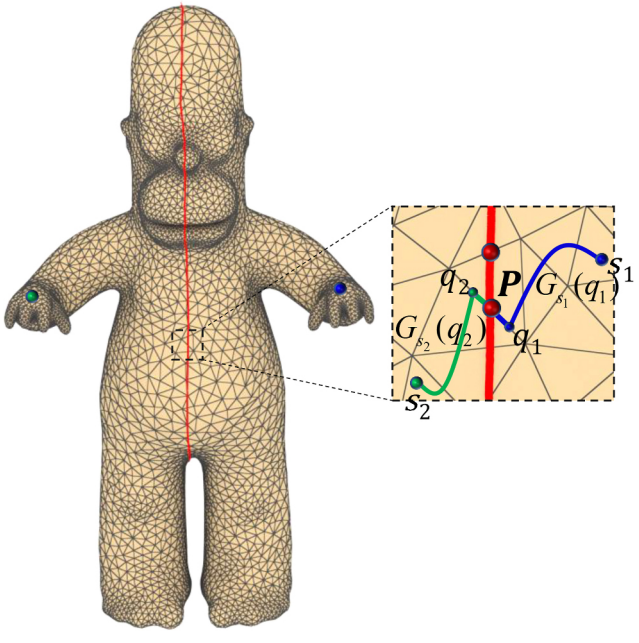


Fig. 6 Intersections between triangle edges and GVD boundaries. Red curve: GVD boundary of sources s_1 and s_2. In the close-up (right), the critical point p is the intersection between a triangle edge and a GVD boundary satisfying $G_{s_1}(q_1) + \|q_1 p\| = G_{s_2}(q_2) + \|q_2 p\|$.

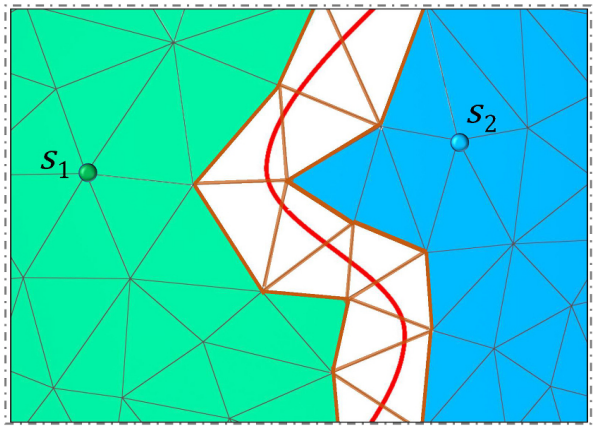


Fig. 7 Redundant primitives include redundant triangles (blue, green) and redundant edges (black). Regions enclosed by yellow edges are those to consider when extracting the geodesic Voronoi diagram.

sources s_i, $i = 1, \cdots, t$, when the Steiner-point graph method terminates, each Steiner point has a geodesic distance $G_{s_i}(q)$ originating from an associated source. The redundant primitives on $\mathcal{M}$ are as follows:

- *Redundant triangle.* A triangle is redundant if all Steiner points on its three edges have the same source;
- *Redundant edge.* An edge is redundant if both adjacent triangles are redundant.
- *Redundant Steiner point.* A Steiner point is redundant if it belongs to a redundant edge.

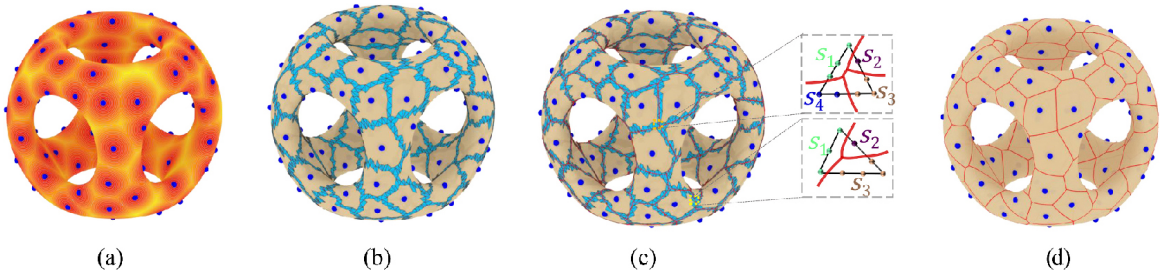


Fig. 8 Algorithm pipeline. (a) Compute multi-source geodesic distance field from 100 randomly sampling generators. (b) During distance propagation, discard redundant primitives and collect the remainder (blue), allowing efficient GVD extraction. (c) Construct the local Apollonius diagram and extract the GVD structure triangle by triangle on the remaining (blue) regions. (d) The output approximate GVD is quickly constructed with high quality.

Using these definitions to discard redundant primitives, we need only process a few remaining triangles of mesh $\mathcal{M}$ to efficiently extract the geodesic Voronoi diagram.

5 Implementation

5.1 Basis

Based on the ideas in Section 4, we present an efficient and practical algorithm to construct geodesic Voronoi diagrams for point source sites. The input is a triangular mesh $\mathcal{M}$ and a set of sites $S = \{s_1, \cdots, s_t\}$. The output is the approximate GVD (AGVD) for $\mathcal{M}$. To provide the multi-source geodesic distance field used in constructing the GVD, we modify Meng et al.'s method [46]. We use a priority queue $\mathcal{Q}$ to organize the Steiner points, which represent wavefronts in the propagation process.

5.2 GVD edge extraction

We discard redundant primitives during distance propagation in the Steiner-point graph method. During each distance iteration, the geodesic computation method propagates the distances of Steiner points using the least geodesic distance. Therefore, the distance of the Steiner point at the head of the priority queue $\mathcal{Q}$ does not decrease. If the edges of a non-discarded triangle $\triangle ABC$ have both distances shorter than the distance of the point at the head of the queue, we can conclude that $\triangle ABC$ is behind the geodesic wavefront; all Steiner points behind the wavefront are no longer updated. We say such triangles are inactive. Thus, we check whether or not $\triangle ABC$ contains GVD edges.

In exact geodesic algorithms, there are $O(n)$ windows on each edge of $\triangle ABC$. Computation of the exact distance from the source to each edge is time-consuming. In our practical implementation, we use an upper bound of the distance from the source s to each edge e [24]. The upper bound distance $d(s, e)$ is $(d(a)+d(b)+\|e\|)/2$, where $d(a), d(b)$ are the geodesic distances from the source to the endpoints of edge e and $\|e\|$ is its length.

Frequently checking for inactive triangles can extract GVD edges in early stages and consume lower memory. However, it has a significant influence on time. In order to balance memory use and time, we check for inactive triangles every λn (an integer) distance iterations, where n is the number of triangles in $\mathcal{M}$: smaller λ results in more frequent checks. After every λn distance iterations, we construct the local Apollonius diagram and extract GVD edges from non-discarded triangles in the inactive region.

5.3 GVD construction

For the Steiner points on opposite edges of site s_i in its 1-ring neighbourhood, we initially assign distances and push them into a priority queue $\mathcal{Q}$. Then, our algorithm progressively computes geodesic distances and extracts GVD structures by performing the following steps in each distance iteration.

(1) Take the Steiner point p from the head of $\mathcal{Q}$ and propagate from p to Steiner points on its adjacent triangles.

(2) Find the adjacent Steiner points which have shorter distances via p and insert them into the priority queue $\mathcal{Q}$.

(3) If the number of distance iterations is an integral multiple of λn or the priority queue $\mathcal{Q}$ is empty, find non-discarded triangles to construct the local Apollonius diagram, and extract GVD edges.

In order to robustly extract GVD edges, besides Steiner points on each retained (non-discarded)

triangle, the surrounding inserted Steiner points of target triangles are also added into the construction process to build joint local Apollonius diagrams, i.e., Steiner points in the 1-ring neighbourhood of retained triangle $\mathcal{F}$ are employed as relay points to partition and extract GVD edges from $\mathcal{F}$.

Algorithm 1 gives further details of GVD construction.

5.4 Complexity analysis

We now analyze the time and space complexity of our algorithm to demonstrate its theoretical efficiency.

Property 5.1 *For GVD construction on an n-face triangle mesh $\mathcal{M}$ with t sites and m Steiner points inserted on each edge, our algorithm has $O(mn\log n)$ time complexity and $O(mn)$ space complexity.*

Proof. In Meng et al.'s Steiner-insertion method [46], $O(m)$ Steiner points are inserted into each triangle edge. Therefore, at most $O(mn)$ Steiner points are pushed onto the priority queue $\mathcal{Q}$ for distance propagation. Sorting them in $\mathcal{Q}$ takes $O(mn\log(mn))$ time and $O(mn)$ space. During GVD extraction, we use the plane sweep algorithm [63], which takes $O(m\log m)$ to construct a local Apollonius diagram. Thus the total time for GVD extraction is $O(mn\log m)$. Overall, the time complexity of our algorithm is bounded by $O(mn\log(mn)+mn\log m) = O(mn\log n)$, while the space complexity is bounded by $O(mn)$. This completes the proof. □

Algorithm 1 Construction of approximate geodesic voronoi diagram for point sources

Input: A mesh $\mathcal{M}$ with n triangles;
A set of sites $S = \{s_1, \cdots, s_t\}$ on $\mathcal{M}$;
Time versus memory control λ;
Number of Steiner points to insert on each edge m.
Output: Approximate geodesic Voronoi diagram.
Create an empty priority queue $\mathcal{Q}$ to hold windows on the wavefront.
for each site s_i in S **do**
  Push the Steiner points on opposite edges of site s_i in its 1-ring neighbourhood into $\mathcal{Q}$.
end for
Set iteration counter $i = 1$
while $\mathcal{Q}$ is nonempty **do**
  Pop the Steiner point p from the head of $\mathcal{Q}$;
  Propagate from p to Steiner points on its adjacent triangles;
  Find adjacent Steiner points which have shorter distances via p and push them into $\mathcal{Q}$.
  if (i mod λn) $= 0$ or $\mathcal{Q}$ is empty **then**
    Traverse inactive regions to find retained triangles
    Construct the local Apollonius diagram and extract GVD edges using these triangles.
  end if
  i++
end while

Previous GVD methods [13, 14, 24] based on the exact geodesic metric require $O(n^2\log n)$ time and $O(n^2)$ space to construct the GVD. Our method has an obvious theoretical advantage in both time and space. Although our GVD is constructed using approximate geodesic distances, we can get a result close to the exact result by inserting Steiner points into triangle edges. We evaluate our algorithm in Section 6.

6 Experiments

6.1 Setting

We implemented our algorithm in Microsoft Visual C++ 2015 without using additional numeric packages. All experiments were conducted on a computer with a 3.60 GHz Intel i7-9700K CPU and 8 GB memory. We evaluated our method on both 3D models widely used in the graphics community, and the Thingi10k 3D shape repository [64], which contains a large number of man-made anisotropic meshes and some have an extremely high degree of anisotropy.

6.2 Error evaluation

Our GVD results are based on approximate geodesic distances, and we can balance accuracy and time required by varying the number of Steiner points inserted into triangle edges. Here, we employ Fréchet distance to assess error in the constructed GVD; it is widely used to measure the similarity of polylines [65, 66]. This distance is also known as dog-leash distance, as it can be intuitively explained in terms of the shortest leash needed when a person and their dog walk at varying speeds along the respective curves.

Converting the curves into polylines is a typical approach [67] for computing the Fréchet distance between arbitrary curves. The method is based on all pairings between the endpoints of line segments on each of the polygonal curves, and uses the paired distances to approximate Fréchet distance, as follows. Let polygonal curve C in $\mathcal{R}^3$ be a continuous function: $[0, n] \to \mathcal{R}^3$. Let $\sigma(C)$ denote the

sequence $(C(0), \cdots, C(n))$ of endpoints on polygonal curve C. Let P and Q be polygonal curves, with corresponding sequences $\sigma(P) = (u_1, \cdots, u_p)$ and $\sigma(Q) = (v_1, \cdots, v_q)$. A coupling L between P and Q is a sequence $(u_{a_1}, v_{b_1}), \cdots, (u_{a_m}, v_{b_m})$ of distinct pairs from $\sigma(P) \times \sigma(Q)$. The length $\|L\|$ of the coupling L is the length of the longest link in L:

$$\|L\| = \max_{i=1,\cdots,m} d(u_{a_i}, v_{b_i}) \tag{7}$$

The discrete Fréchet distance between curves P and Q is then defined as

$$\mathcal{F}(P, Q) = \min\{\|L\| \,|\, L \text{ is a coupling between } P \text{ and } Q\}$$

Let us consider the GVD on a genus-r ($r \geqslant 0$) mesh $\mathcal{M}$ with sites $S = \{s_1, \cdots, s_t\}$, $t \geqslant 2$, and assume each cell $GC(s_i)$ consists of a set of closed curves $C = \{c_1, \cdots, c_k\}$, we define the Fréchet distance of cell $GC(s_i)$ to be $\sum_{j=1}^{k} \mathcal{F}(c_j, \widetilde{c_j})$ where c_j is the curve for the exact GVD Voronoi cell and $\widetilde{c_j}$ is the curve of our GVD Voronoi cell. We further define the Fréchet distance between the exact GVD and our $\widetilde{GVD}$ on mesh $\mathcal{M}$ with t sites as

$$\mathcal{F}(GVD, \widetilde{GVD}) = \sum_{i=1}^{t} \sum_{j=1}^{k} \mathcal{F}(c_j^i, \widetilde{c_j^i}) \tag{8}$$

In order to evaluate the approximate GVD results more intuitively, we define the error function $\delta(\widetilde{GVD})$ as

$$\delta(\widetilde{GVD}) = 1 - \exp\left(-\frac{\mathcal{F}(GVD, \widetilde{GVD})}{c^2}\right) \tag{9}$$

This error measure depends on the choice of the constant c. If it is chosen too small or too large, the function poorly describes GVD error. Our extensive experiments show that the best choice of $c \approx 10$. Obviously, $\delta(\widetilde{GVD})$ is in $[0, 1]$ where 0 indicates that our GVD and the exact GVD result are identical and 1 means they are completely different.

In order to construct GVD, we use the Steiner point insertion method [46] to acquire the geodesic distance field. Errors in geodesic distances are controlled by inserted Steiner points on triangle edges which indirectly influence the accuracy of the AGVD construction. Intuitively, the more inserted Steiner points on each edge, the higher expected accuracy of the AGVD result. Our method can balance the time, memory, and accuracy well by varying the number of inserted Steiner points.

Taking the 50k-face Cow model as an example in Fig. 9, we place 30 generators on the model and insert 5, 8, and 12 Steiner points on each edge to construct the AGVD. As Fig. 9 shows, the AGVD errors rapidly decrease with increasing Steiner points. With 12 Steiner points, our algorithm only takes 0.24 s and 1.32 MB memory to construct an AGVD with 0.07% error, while the exact GVD method [14] takes 0.59 s and 2.79 MB memory. Generally, we set $m = 8$ as the default number of Steiner points on each edge in our experimental tests.

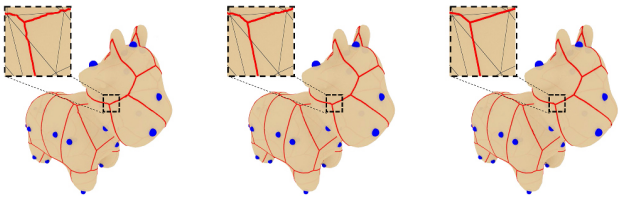


Fig. 9 GVD construction on the Cow model with 30 generators, inserting different numbers of Steiner points on each triangle edge. Left to right: using 5, 8, and 12 Steiner leads to respective GVD errors of 1.24%, 0.31%, and 0.07%.

Our algorithm uses relatively little time and memory to construct an approximate GVD with an extremely low error close to the exact GVD result. This has an obvious advantage in error insensitive applications, e.g., geodesic remeshing, which we consider in Section 7. Without a doubt, higher accuracy must require more computational cost. However, our algorithm can achieve much higher accuracy at the cost of a small extra computational amount. As Fig. 10 shows, when m is set to 10, the

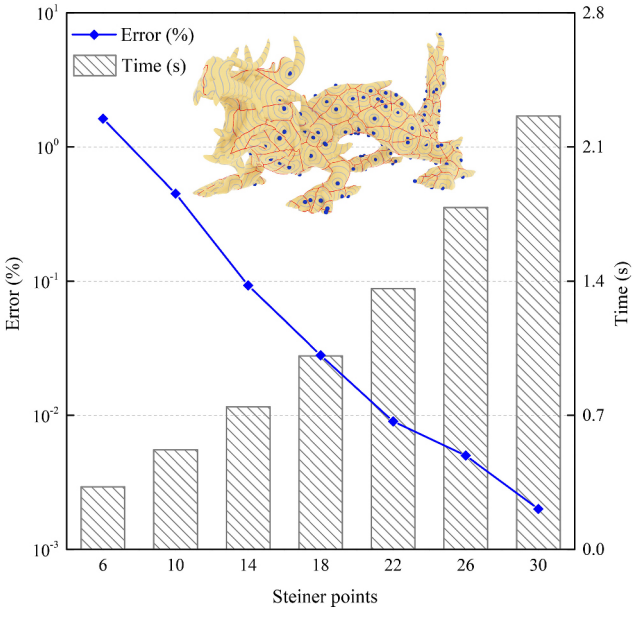


Fig. 10 Time and accuracy with respect to the number of Steiner points inserted into each mesh edge. Inserting more points gives higher accuracy, but takes more time.

time for GVD construction is 0.51 s, and when m increases to 30, the time used is only 2.26 s. Table 1 gives further experimental results to emphasize the balance between time, memory, and accuracy.

6.3 Performance

6.3.1 Assessment

We use running time and peak memory to compare performance between our method and other leading methods. There are many factors which influence the performance of GVD construction methods. Here, we select the three most significant quantities: number m of Steiner points per triangle edge, number t of source sites, control parameter λ balancing time and memory. Since choice of number m of Steiner points was discussed in Section 6.2, we concentrate on the other two factors here.

6.3.2 Number of source sites

We first consider the effect of increasing the number of source sites, using various models. We used 2–2000 source sites and measured AVGD time and memory consumption. Results are shown in Fig. 11 and Table 2.

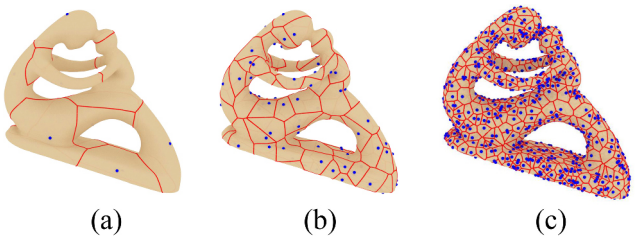


Fig. 11 GVD construction for the 30k-face Fertility model with different numbers of generators. (a) Using 10 generators requires 0.186 s and 0.957 MB peak memory. (b) Using 100 generators takes 0.126 s and 0.659 MB peak memory. (c) Using 1000 generators uses 0.113 s and 0.521 MB peak memory.

As Table 2 shows, the time needed for GVD construction drops with an increasing number of generators, since each geodesic wavefront just covers a small region of the model surface. Furthermore, more time is needed because a relatively large t implies that

Table 1 Characterisation of various GVD construction methods, using 100 generators for various models. m is the number of Steiner points per edge

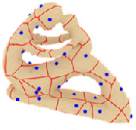
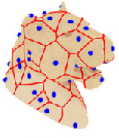
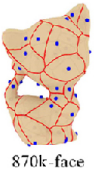
Method / Model	Assessment	 30k-face	 120k-face	 300k-face	 870k-face
Ours ($m = 5$)	Time (s)	**0.095**	**0.282**	**0.455**	**2.293**
	Error (%)	1.570	1.393	1.407	1.126
	Peak memory (MB)	**0.423**	**1.032**	**1.724**	**3.483**
Ours ($m = 8$)	Time (s)	0.126	0.438	0.797	3.432
	Error (%)	0.517	0.431	0.489	0.332
	Peak memory (MB)	0.659	1.643	2.959	4.137
Ours ($m = 12$)	Time (s)	0.176	0.669	1.147	6.084
	Error (%)	**0.073**	**0.066**	**0.069**	**0.051**
	Peak memory (MB)	1.062	2.413	4.160	7.126
Ref. [13]	Time (s)	0.775	3.297	9.425	74.640
	Error (%)	—	—	—	—
	Peak memory (MB)	29.130	148.383	371.589	1638.220
Ref. [24] ($c = 1$)	Time (s)	1.045	5.437	12.497	45.168
	Error (%)	—	—	—	—
	Peak memory (MB)	91.061	291.038	503.928	1901.827
Ref. [14]	Time (s)	0.361	1.474	2.511	13.385
	Error (%)	—	—	—	—
	Peak memory (MB)	1.912	4.541	7.072	12.827
Ref. [37]	Time (s)	0.294	1.218	1.911	10.774
	Error (%)	8.334	7.460	7.861	6.482
	Peak memory (MB)	1.307	3.135	5.440	11.623

Table 2 Relationship between the number of generators, and time and memory requirements. We use the default number of Steiner points $m = 8$ on each edge

Generators	Fertility (30k)		Lion head (120k)	
	Time (s)	Memory (MB)	Time (s)	Memory (MB)
2	0.194	1.086	0.441	2.074
4	0.197	1.053	0.443	1.991
8	0.203	1.010	0.447	1.952
10	0.186	0.957	0.452	1.917
50	0.157	0.821	0.494	1.854
100	0.126	0.659	0.438	1.643
500	0.109	0.557	0.368	1.391
1000	0.113	0.521	0.355	1.325
1500	0.117	0.497	0.343	1.274
2000	0.124	0.481	0.336	1.213

the local Apollonius diagram is more complicated, so takes longer to compute.

6.3.3 Balancing time and memory

Our algorithm considers inactive triangles after every λn distance iterations where λ controls the balance between time and memory. Smaller λ results in more frequently checking the inactive triangles, so extracting GVD edges earlier and consuming less memory. However, this has a significant effect on time taken. Using a larger λ results in less frequent checking of the inactive triangles, which is faster but consumes more memory. We verify these claims and show the effect of changing λ in Fig. 12. We set $\lambda = 1.0$ by default value in our experimental tests.

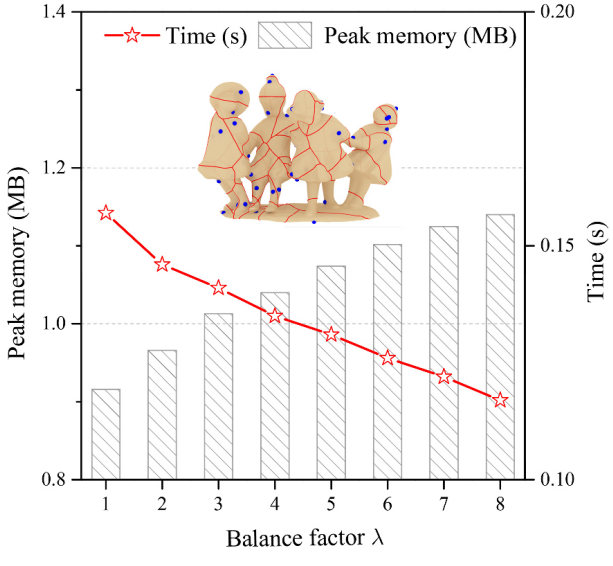


Fig. 12 Balancing time and memory, using λ. We employ the 28k-face Kids model to construct a GVD with 50 sites and vary λ to control the balance between time taken and memory usage.

6.4 Qualitative comparison

In this section, we make a qualitative comparison between our method and the four typical methods most in common with our method.

6.4.1 Comparison to Ref. [13]

Liu et al. studied the analytic structure of bisectors and Voronoi diagrams on triangulated surfaces, and proposed a practical algorithm to construct the Voronoi diagram on a triangulated mesh $\mathcal{M}$. They used the MMP method [22] to obtain geodesic information for GVD construction. Since the distance field on an edge e of $\mathcal{M}$ can have $O(n)$ extrema, they partition e into sub-edges such that the distance field value on each sub-edge is monotone and linear. Compared to Liu et al.'s method, our method has two significant advantages. On one hand, their method needs to subdivide mesh triangles until the bisectors cross each face at most once. As a result, the method is sensitive to the mesh resolution while our method is robust on meshes of the same shape with different resolutions. See Section 6.5 for further details. On the other hand, using the Steiner-point graph method, we can get GVD results which are close to exact, using less time and memory. See Table 1.

6.4.2 Comparison to Ref. [24]

Unlike most previous GVD algorithms, this approach uses polyline generators. They revealed that a typical GVD bisector contains linear, hyperbolic, and parabolic segments. To overcome the challenge this presents, a new concept called the local Voronoi diagram (LVD) was introduced, which is a combination of additively weighted Voronoi diagrams and line-segment Voronoi diagrams defined locally in the plane. Our algorithm extracts GVD edges in a similar manner to this method. The noticeable difference is that they use the MMP framework to compute exact geodesic distances, while our algorithm relies on approximate geodesics computed by the Steiner point insertion method. Our relative advantage is that we can flexibly balance the time, memory, and error during GVD construction; this is an obvious advantage in error insensitive applications, as shown in Section 7.

6.4.3 Comparison to Ref. [14]

This paper proposed an efficient algorithm to construct the GVD, aiming to reduce redundant computation to save time and memory. A redundant

window removal process is used during GVD construction: in the window-VTP algorithm it selectively retains windows on edges. The key issue is how to detect and remove redundant windows simultaneously with geodesic wavefront propagation. Compared with this method, our method is easier to implement and more computationally efficient. Using the 50k-face Monster model in a test, our method is about three times faster for an error of less than 0.3% (see Fig. 1). Meanwhile, our peak memory usage is about one third of that of Qin et al.'s method.

6.4.4 Comparison to Ref. [37]

In this paper, Xin et al. presented an approximate algorithm to construct the GVD using point generators. They first computed multisource geodesic distances using generators on the target mesh. Upon termination of this phase, each vertex is labelled with its nearest source. They then construct bisectors by checking triangles containing at least two intersection points. If there are two intersection points, a segment is used to connect points. If there are three intersection points, a point is found inside the triangle and connected to all three intersections. Finally, the GVD is constructed by tracing the bisectors. The method is easy to implement and works well for high quality and resolution meshes. However, it may produce low quality results when the input meshes are poor, e.g., have irregular triangulations or sharp corners. The natural solution is to improve the triangulation quality or increase mesh resolution. However, remeshing and subdivision operations are time-consuming and may have their own issues. Compared to this method, our algorithm is robust to mesh resolution and triangulation quality, as well as providing an efficient way to construct an approximate GVD. We next discuss robustness of our method further.

6.5 Robustness

Our algorithm is robust to changes in mesh resolution and triangulation quality. In order to demonstrate robustness, we constructed the AGVD on a triangular mesh with different resolutions and varying quality. The Lo value [68] provides a general way to evaluate mesh quality; higher quality meshes have values closer to 1. AGVD results on the Lucy model at three different resolutions and three different qualities are shown in Fig. 13, indicating that our algorithm can construct reliable and relatively consistent AGVD results.

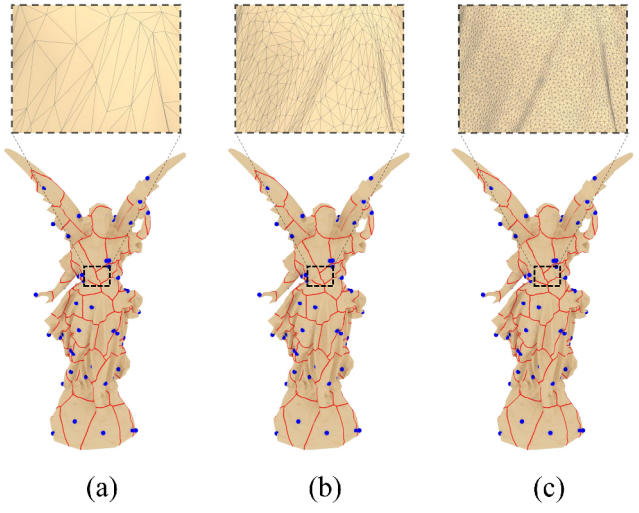


Fig. 13 Our method is insensitive to input mesh quality and resolution: (a) 5k-face, Lo = 0.14; (b) 50k-face, Lo = 0.38; (c) 500k-face, Lo = 0.87.

We also examined our method under varying mesh anisotropy. We use the anisotropy measure defined in Xu et al. [31] for a triangular face f:

$$\tau(f) = \frac{P \cdot H}{2\sqrt{3}S}$$

where P, H, and S are respectively its half-perimeter, longest edge length, and area. The mesh anisotropy measure is then computed as the average of $\tau(f)$ over all faces of the mesh. Larger τ indicates a higher degree of anisotropy. We selected two models from the Thingi10k repository with average anisotropy τ_{avg} of 31.1 and 9.9, and maximal anisotropy τ_{max} of 1062.3 and 25,183.7, respectively. Figure 14 shows our AGVD results for 50 sites on these highly anisotropic meshes.

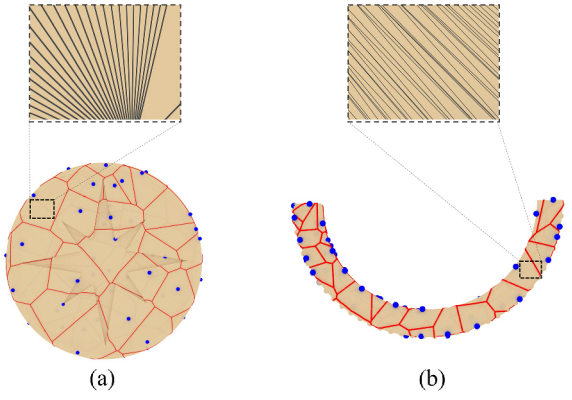


Fig. 14 Our algorithm produces accurate GVD results for meshes with highly anisotropic triangles (average anisotropy $\tau_{\text{avg}} = 31.1, 9.9$, respectively). Using 12 Steiner points per edge, AGVD errors are 0.42% and 0.38%, demonstrating that our method copes well with anisotropic triangles.

7 Application to remeshing

That the Delaunay triangulation of a point set S is the dual of its Voronoi diagram is well known. Leibon and Letscher [49] showed that, if the sampling points are sufficiently dense, the dual triangulation of the Voronoi diagram on $\mathcal{M}$ exists. This observation has an application to remeshing the compact triangular models reconstructed from laser scanning data. In detail, we first compute a geodesic Voronoi diagram (GVD) by selecting sources as generators and then find its dual graph as the remeshed model $\tilde{\mathcal{M}}$. In this process, we can save time and memory by constructing an AGVD close to the exact result, making a new mesh more efficiently than related methods [13, 14]. As a demonstration, we remesh the 200k-face Centaur model using 2000 randomly selected sources. Liu et al.'s method [13] takes 19.72 s and 549.51 MB memory while Qin et al.'s method [14] takes 6.43 s and 24.01 MB memory. Our method runs faster and uses less memory, taking 2.12 s and using 7.83 MB of memory. See Fig. 15.

8 Conclusions and future work

In this paper, we have presented an efficient and practical algorithm to construct geodesic Voronoi diagrams with point source sites. Unlike previous methods that depend on the exact geodesic metric to construct GVD, we reformulate a Steiner point insertion method [46], to obtain geodesic distances effectively. This substitution reduces GVD computation time and memory usage. We utilize local Apollonius diagrams with weighted Steiner points to partition mesh triangles and extract hyperbolic and linear segments for encoding the complicated GVD bisector structures. In order to balance memory and time requirements, we only process significant triangles instead of using a brute force search over all triangles. Every λn distance iterations, where n is the number of triangles in $\mathcal{M}$, we construct the Apollonius diagram and extract GVD edges. We also suggest an evaluation measure based on discrete Fréchet distance to assess similarity between our result and the exact GVD result. Although our GVD results are based on approximate geodesic distances, we get GVD results close to exact results by inserting Steiner points on triangle edges.

In the future, we hope to enhance our algorithm so that it iteratively inserts Steiner points until a pre-specified error bound is reached. We also intend to investigate other applications that would benefit from approximate geodesic Voronoi diagrams, such as tree skeleton extraction and classification, point pattern analysis on a mesh [13], and so on.

Acknowledgements

The authors would like to thank the reviewers for their valuable suggestions. This work was supported in part by the Youth Teacher Development Foundation of Harbin Institute of Technology (IDGA10002143), the National Natural Science Foundation of China (62072139, 62272277, 62072284), the National Key

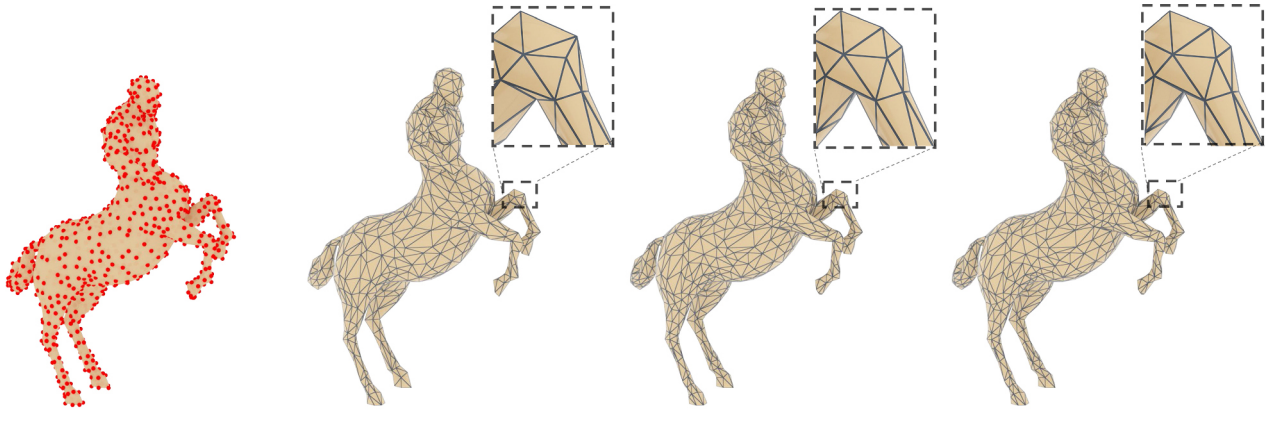


Fig. 15 Remeshing using our algorithm and other leading methods to remesh the 200k-face Centaur model. (a) 2000 samples points (red) on the original mesh. (b–d) Remeshing results from our method and others.

R&D Program of China (2021YFB1715900), and the Joint Funds of the National Natural Science Foundation of China (U22A2033).

Declaration of competing interest

The authors have no competing interests to declare that are relevant to the content of this article.

Electronic Supplementary Material

Supplementary material is available in the online version of this article at `https://doi.org/10.1007/s41095-022-0326-0`.

References

[1] Tsai, J.; Gerstein, M.; Levitt, M. Simulating the minimum core for hydrophobic collapse in globular proteins. *Protein Science* Vol. 6, No. 12, 2606–2616, 1997.

[2] Liu, Y. J.; Yu, M. J.; Li, B. J.; He, Y. Intrinsic manifold SLIC: A simple and efficient method for computing content-sensitive superpixels. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 40, No. 3, 653–666, 2018.

[3] Dong, X.; Chen, Z. G.; Liu, Y. J.; Yao, J. F.; Guo, X. H. GPU-based supervoxel generation with a novel anisotropic metric. *IEEE Transactions on Image Processing* Vol. 30, 8847–8860, 2021.

[4] Liu, Y.; Wang, W. P.; Lévy, B.; Sun, F.; Yan, D. M.; Lu, L.; Yang, C. On centroidal Voronoi tessellation—Energy smoothness and fast computation. *ACM Transactions on Graphics* Vol. 28, No. 4, Article No. 101, 2009.

[5] Liu, Y. J.; Xu, C. X.; Yi, R.; Fan, D.; He, Y. Manifold differential evolution (MDE). *ACM Transactions on Graphics* Vol. 35, No. 6, Article No. 243, 2016.

[6] Wang, X. N.; Ying, X.; Liu, Y. J.; Xin, S. Q.; Wang, W. P.; Gu, X. F.; Mueller-Wittig, W.; He, Y. Intrinsic computation of centroidal voronoi tessellation (CVT) on meshes. *Computer-Aided Design* Vol. 58, 51–61, 2015.

[7] Stanković T.; Shea, K. Investigation of a Voronoi diagram representation for the computational design of additively manufactured discrete lattice structures. *Journal of Mechanical Design* Vol. 142, No. 11, 111704, 2020.

[8] Dai, G. Y.; Lv, H. X.; Chen, L. Y.; Zhou, B. B.; Xu, P. A novel coverage holes discovery algorithm based on Voronoi diagram in wireless sensor networks. *International Journal of Hybrid Information Technology* Vol. 9, No. 3, 273–282, 2016.

[9] Boissonnat, J. D.; Wormser, C.; Yvinec, M. Curved Voronoi diagrams. In: *Effective Computational Geometry for Curves and Surfaces*. Berlin Heidelberg: Springer, 67–116, 2006.

[10] Aurenhammer, F. Voronoi diagrams—A survey of a fundamental geometric data structure. *ACM Computing Surveys* Vol. 23, No. 3, 345–405, 1991.

[11] Liu, J.; Liu, S. A survey on applications of Voronoi diagrams. *Journal of Engineering Graphics* Vol. 22, No. 2, 125–132, 2004.

[12] Kunze, R.; Wolter, F. E.; Rausch, T. Geodesic Voronoi diagrams on parametric surfaces. *Proceedings Computer Graphics International* Vol. 16, No. 3, 230–237, 1997.

[13] Liu, Y. J.; Chen, Z. Q.; Tang, K. Construction of iso-contours, bisectors, and Voronoi diagrams on triangulated surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 33, No. 8, 1502–1517, 2011.

[14] Qin, Y. P.; Yu, H. C.; Zhang, J. J. Fast and memory-efficient Voronoi diagram construction on triangle meshes. *Computer Graphics Forum* Vol. 36, No. 5, 93–104, 2017.

[15] Na, H. S.; Lee, C. N.; Cheong, O. Voronoi diagrams on the sphere. *Computational Geometry* Vol. 23, No. 2, 183–194, 2002.

[16] Onishi, K.; Takayama, N. Construction of Voronoi diagram on the upper half-plane. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* Vol. E79-A, No. 4, 533–539, 1996.

[17] Medimegh, N.; Belaid, S.; Werghi, N. A survey of the 3D triangular mesh watermarking techniques. *International Journal of Multimedia* Vol. 1, No. 1, 33–39, 2015.

[18] Peyré, G.; Cohen, L. D. Geodesic remeshing using front propagation. *International Journal of Computer Vision* Vol. 69, No. 1, 145–156, 2006.

[19] Peethambaran, J.; Muthuganapathy, R. Reconstruction of water-tight surfaces through Delaunay sculpting. *Computer-Aided Design* Vol. 58, 62–72, 2015.

[20] Kimmel, R.; Kiryati, N.; Bruckstein, A. M. Multivalued distance maps for motion planning on surfaces with moving obstacles. *IEEE Transactions on Robotics and Automation* Vol. 14, No. 3, 427–436, 1998.

[21] Lu, L.; Lévy, B.; Wang, W. P. Centroidal Voronoi tessellation of line segments and graphs. *Computer Graphics Forum* Vol. 31, No. 2pt4, 775–784, 2012.

[22] Mitchell, J. S. B.; Mount, D. M.; Papadimitriou, C. H. The discrete geodesic problem. *SIAM Journal on Computing* Vol. 16, No. 4, 647–668, 1987.

[23] Qin, Y. P.; Han, X. G.; Yu, H. C.; Yu, Y. Z.; Zhang, J. J. Fast and exact discrete geodesic computation based on triangle-oriented wavefront propagation. *ACM Transactions on Graphics* Vol. 35, No. 4, Article No. 125, 2016.

[24] Xu, C. X.; Liu, Y. J.; Sun, Q.; Li, J. Y.; He, Y. Polyline-sourced geodesic Voronoi diagrams on triangle meshes. *Computer Graphics Forum* Vol. 33, No. 7, 161–170, 2014.

[25] Bose, P.; Maheshwari, A.; Shu, C.; Wuhrer, S. A survey of geodesic paths on 3D surfaces. *Computational Geometry* Vol. 44, No. 9, 486–498, 2011.

[26] Crane, K.; Livesu, M.; Puppo, E.; Qin, Y. P. A survey of algorithms for geodesic paths and distances. *arXiv preprint* arXiv:2007.10430, 2020.

[27] Surazhsky, V.; Surazhsky, T.; Kirsanov, D.; Gortler, S. J.; Hoppe, H. Fast exact and approximate geodesics on meshes. *ACM Transactions on Graphics* Vol. 24, No. 3, 553–560, 2005.

[28] Chen, J. D.; Han, Y. J. Shortest paths on a polyhedron. In: Proceedings of the 6th Annual Symposium on Computational Geometry, 360–369, 1990.

[29] Xin, S. Q.; Wang, G. J. Improving Chen and Han's algorithm on the discrete geodesic problem. *ACM Transactions on Graphics* Vol. 28, No. 4, Article No. 104, 2009.

[30] Ying, X.; Xin, S. Q.; He, Y. Parallel Chen–Han (PCH) algorithm for discrete geodesics. *ACM Transactions on Graphics* Vol. 33, No. 1, Article No. 9, 2014.

[31] Xu, C. X.; Wang, T. Y.; Liu, Y. J.; Liu, L. G.; He, Y. Fast wavefront propagation (FWP) for computing exact geodesic distances on meshes. *IEEE Transactions on Visualization and Computer Graphics* Vol. 21, No. 7, 822–834, 2015.

[32] Du, J.; He, Y.; Fang, Z.; Meng, W. L.; Xin, S. Q. On the vertex-oriented triangle propagation (VTP) algorithm: Parallelization and approximation. *Computer-Aided Design* Vol. 130, 102943, 2021.

[33] Kimmel, R.; Sethian, J. A. Computing geodesic paths on manifolds. *Proceedings of the National Academy of Sciences of the United States of America* Vol. 95, No. 15, 8431–8435, 1998.

[34] Weber, O.; Devir, Y. S.; Bronstein, A. M.; Bronstein, M. M.; Kimmel, R. Parallel algorithms for approximation of distance maps on parametric surfaces. *ACM Transactions on Graphics* Vol. 27, No. 4, Article No. 104, 2008.

[35] Crane, K.; Weischedel, C.; Wardetzky, M. Geodesics in heat. *ACM Transactions on Graphics* Vol. 32, No. 5, Article No. 152, 2013.

[36] Solomon, J.; Rustamov, R.; Guibas, L.; Butscher, A. Earth mover's distances on discrete surfaces. *ACM Transactions on Graphics* Vol. 33, No. 4, Article No. 67, 2014.

[37] Xin, S. Q.; Ying, X.; He, Y. Constant-time all-pairs geodesic distance query on triangle meshes. In: Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, 31–38, 2012.

[38] Ying, X.; Wang, X. N.; He, Y. Saddle vertex graph (SVG). *ACM Transactions on Graphics* Vol. 32, No. 6, Article No. 170, 2013.

[39] Lanthier, M.; Maheshwari, A.; Sack, J.-R. Approximating shortest paths on weighted polyhedral surfaces. *Algorithmica* Vol. 30, No. 4, 527–562, 2001.

[40] Lanthier, M.; Maheshwari, A.; Sack, J. R. Approximating weighted shortest paths on polyhedral surfaces. In: Proceedings of the 13th Annual Symposium on Computational Geometry, 274–283, 1997.

[41] Aleksandrov, L.; Lanthier, M.; Maheshwari, A.; Sack, J. R. An ε-approximation algorithm for weighted shortest paths on polyhedral surfaces. In: *Algorithm Theory — SWAT'98. Lecture Notes in Computer Science, Vol. 1432*. Arnborg, S.; Ivansson, L. Eds. Springer Berlin Heidelberg, 11–22, 1998.

[42] Aleksandrov, L.; Maheshwari, A.; Sack, J. R. Determining approximate shortest paths on weighted polyhedral surfaces. *Journal of the ACM* Vol. 52, No. 1, 25–53, 2005.

[43] Adikusuma, Y. Y.; Du, J.; Fang, Z.; He, Y. An accuracy controllable and memory efficient method for computing high-quality geodesic distances on triangle meshes. *Computer-Aided Design* Vol. 150, 103333, 2022.

[44] Adikusuma, Y. Y.; Fang, Z.; He, Y. Fast construction of discrete geodesic graphs. *ACM Transactions on Graphics* Vol. 39, No. 2, Article No. 14, 2020.

[45] Aleksandrov, L.; Maheshwari, A.; Sack, J. R. Approximation algorithms for geometric shortest path problems. In: Proceedings of the 32nd Annual ACM Symposium on Theory of Computing, 286–295, 2000.

[46] Meng, W. L.; Xin, S. Q.; Tu, C. H.; Chen, S. M.; He, Y.; Wang, W. P. Geodesic tracks: Computing discrete geodesics with track-based Steiner point propagation. *IEEE Transactions on Visualization and Computer Graphics* Vol. 28, No. 12, 4887–4901, 2022.

[47] Lee, D. T. Medial axis transformation of a planar shape. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. PAMI-4, No. 4, 363–369, 1982.

[48] Giblin, P.; Kimia, B. B. A formal classification of 3D medial axis points and their local geometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 26, No. 2, 238–251, 2004.

[49] Leibon, G.; Letscher, D. Delaunay triangulations and Voronoi diagrams for Riemannian manifolds. In: Proceedings of the 16th Annual Symposium on Computational Geometry, 341–349, 2000.

[50] Boissonnat, J. D.; Dyer, R.; Ghosh, A. Constructing intrinsic Delaunay triangulations of submanifolds. *arXiv preprint* arXiv:1303.6493, 2013.

[51] Augenbaum, J. M.; Peskin, C. S. On the construction of the Voronoi mesh on a sphere. *Journal of Computational Physics* Vol. 59, No. 2, 177–192, 1985.

[52] Senechal, M. Spatial tessellations: Concepts and applications of Voronoi diagrams. *Science* Vol. 260, No. 5111, 1170–1173, 1993.

[53] Kimmel, R.; Sethian, J. A. Fast Voronoi diagrams and offsets on triangulated surfaces. Technical Report. Technion-Israel Inst of Tech Haifa Dept of Computer Science, 2000.

[54] Liu, Y. J.; Tang, K. The complexity of geodesic Voronoi diagrams on triangulated 2-manifold surfaces. *Information Processing Letters* Vol. 113, No. 4, 132–136, 2013.

[55] Liu, Y. J.; Fan, D.; Xu, C. X.; He, Y. Constructing intrinsic Delaunay triangulations from the dual of geodesic Voronoi diagrams. *ACM Transactions on Graphics* Vol. 36, No. 2, Article No. 15, 2017.

[56] Van Kreveld, M.; Schwarzkopf, O.; de Berg, M.; Overmars, M. Computational geometry algorithms and applications. *Computer Graphics Forum* Vol. 13, No. 3, 12–16, 2000.

[57] Rong, G. D.; Liu, Y.; Wang, W. P.; Yin, X. T.; Gu, D.; Guo, X. H. GPU-assisted computation of centroidal Voronoi tessellation. *IEEE Transactions on Visualization and Computer Graphics* Vol. 17, No. 3, 345–356, 2011.

[58] Aurenhammer, F. Power diagrams: Properties, algorithms and applications. *SIAM Journal on Computing* Vol. 16, No. 1, 78–96, 1987.

[59] Gavrilova, M.; Rokne, J. An efficient algorithm for construction of the power diagram from the Voronoi diagram in the plane. *International Journal of Computer Mathematics* Vol. 61, Nos. 1–2, 49–61, 1996.

[60] Karavelas, M. I.; Yvinec, M. Dynamic additively weighted Voronoi diagrams in 2D. In: *Algorithms — ESA 2002. Lecture Notes in Computer Science, Vol. 2461*. Möhring, R.; Raman, R. Eds. Springer Berlin Heidelberg, 586–598, 2002.

[61] Karavelas, M. I.; Emiris, I. Z. Predicates for the planar additively weighted Voronoi diagram. Technical Report ECG-TR-122201-01. INRIA Sophia-Antipolis, 2002.

[62] Wang, P. H.; Yuan, N.; Ma, Y. W.; Xin, S. Q.; He, Y.; Chen, S. M.; Xu, J.; Wang, W. Robust computation of 3D Apollonius diagrams. *Computer Graphics Forum* Vol. 39, No. 7, 43–55, 2020.

[63] Fortune, S. A sweepline algorithm for Voronoi diagrams. *Algorithmica* Vol. 2, Nos. 1–4, 153–174, 1987.

[64] Zhou, Q. N.; Jacobson, A. Thingi10K: A dataset of 10,000 3D-printing models. *arXiv preprint* arXiv:1605.04797, 2016.

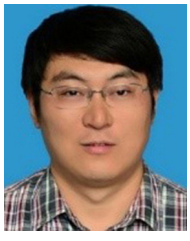
[65] Alt, H.; Godau, M. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications* Vol. 5, Nos. 01n02, 75–91, 1995.

[66] Rote, G. Computing the Fréchet distance between piecewise smooth curves. *Computational Geometry* Vol. 37, No. 3, 162–174, 2007.

[67] Eiter, T.; Mannila, H. Computing discrete Fréchet distance. Technical Report. 1994. Available at http://www.kr.tuwien.ac.at/staff/eiter/et-archive/cdtr9464.pdf.

[68] Lo, S. H. A new mesh generation scheme for arbitrary planar domains. *International Journal for Numerical Methods in Engineering* Vol. 21, No. 8, 1403–1426, 1985.

Wenlong Meng received his Ph.D. degree from Shandong University, in 2022. He is currently a lecturer with the School of Computer Science and Technology, Harbin Institute of Technology, Weihai. His research interests include computational geometry, computer graphics, and computer-aided design.



Pengbo Bo received his B.S. and M.E. degrees in computer science at the School of Computer Science, Shandong University, Jinan, in 2001 and 2004, respectively, and his Ph.D. degree in computer science from the University of Hong Kong, in 2010. He is a professor and doctoral supervisor at the School of Computer Science and Technology, Harbin Institute of Technology, Weihai. His research interests include computer graphics, 3D visual computing, and CNC machining.

Xiaodong Zhang received his M.S. degree in Harbin Institute of Technology. He is currently a lecturer with the School of Computer Science and Technology, Harbin Institute of Technology, Weihai. His research interests include design and analysis of algorithms.

Jixiang Hong is an undergraduate student at Taishan College, Shandong University. He entered the university in 2019. He majors in computer science and technology. His interests are focused in the area of computer graphics.

Shiqing Xin received his Ph.D. degree from Zhejiang University, in 2009. He was a research fellow with Nangyang Technological University, Singapore, for three years. He is currently an associate professor with the School of Computer Science, Shandong University. He has authored or coauthored more than 60 papers in top journals and conferences, including *IEEE Transactions on Visualization* and *Computer Graphics* and *ACM Transactions on Graphics*. His research interests include various geometry processing algorithms, especially geodesic computation approaches and Voronoi/power tessellation methods. He was the recipient of three best paper awards and many other academic awards.

Changhe Tu received his B.Sc., M.Eng., and Ph.D. degrees from Shandong University, in 1990, 1993, and 2003, respectively. He is currently a professor with the School of Computer Science and Technology, Shandong University. He currently leads the CG-VIS Group, Shandong University. He has authored or coauthored more than 100 papers in international journals and conferences. His research interests include computer graphics, 3D vision, and computer-aided geometric design.