GBGVD: Growth-based Geodesic Voronoi Diagrams

Yunjia Qi
Shandong University
Qingdao, China
qiyunjia.ya@gmail.com

Chen Zong
Shandong University
Qingdao, China
zongchen@mail.sdu.edu.cn

Yunxiao Zhang
Shandong University
Qingdao, China
zhangyunxiaox@gmail.com

Shuangmin Chen
Qingdao University of
Science and Technology
Qingdao, China
csmqq@163.com

Minfeng Xu
Shandong University of
Finance and Economics,
Jinan, China
mfxu_sdu@163.com

Lingqiang Ran
Shandong University of
Finance and Economics,
Jinan, China
ranlingqiang@sdufe.edu.cn

Jian Xu
Dalian University
of Technology
Dalian, China
xujian1028@dlut.edu.cn

Shiqing Xin
Shandong University
Qingdao, China
xinshiqing@sdu.edu.cn

Ying He
Nanyang Technological University
Singapore
yhe@ntu.edu.sg

Abstract

Given a set of generators, the geodesic Voronoi diagram (GVD) defines how the base surface is decomposed into separate regions such that each generator dominates a region in terms of geodesic distance to the generators. Generally speaking, each ordinary bisector point of the GVD is determined by two adjacent generators while each branching point of the GVD is given by at least three generators. When there are sufficiently many generators, straight-line distance serves as an effective alternative of geodesic distance for computing GVDs. However, for a set of sparse generators, one has to use exact or approximate geodesic distance instead, which requires a high computational cost to trace the bisectors and the branching points. We observe that it is easier to infer the branching points by stretching the ordinary segments than competing between wavefronts from different directions. Based on the observation, we develop an unfolding technique to compute the ordinary points of the GVD, as well as a growth-based technique to stretch the traced bisector segments such that they finally grow into a complete GVD. Experimental results show that our algorithm runs 3 times as fast as the state-of-the-art method at the same accuracy level.

Keywords: Voronoi diagram, geodesic distances, fast marching method, triangle mesh, computational geometry

1. Introduction

Given a 2-manifold surface as well as a set of generators on the surface, the problem of computing a geodesic Voronoi diagram (GVD) is to find a way of dividing the surface into regions such that each generator owns a corresponding region consisting of all points that are closer to the generator than the others.

The GVD is much like the Voronoi diagram in a finite-dimensional Euclidean space except that the GVD is defined on a curved surface, driven by geodesic distance. It has a variety of applications such as remeshing [13], shape segmentation [15], point pattern analysis [14], and surface reconstruction [20].

The task of finding the Euclidean Voronoi diagrams is to compute the Voronoi vertices as well as the combinatorial structure that connects the Voronoi vertices. However, computing Voronoi diagrams on a 2-manifold surface is much different from the Euclidean situation since each Voronoi edge is constrained to a curved surface. The difficulties are two-fold. On one hand, it is time-consuming to infer the geodesic distances on a polygonal surface that consists of a large number of triangles. On the other hand, it is non-trivial to predict how the GVD structure passes through triangles based on the competition between wavefronts from different directions.

Leibon and Letscher [12] proved that when the generators are sufficiently dense, the local part of the GVD on a 2-manifold surface is conformal to the Voronoi diagram in

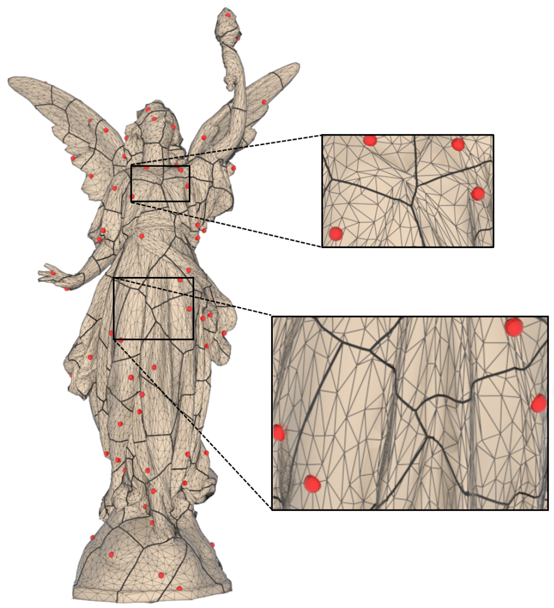


Figure 1. The GVD produced by our algorithm on the 20K-face Lucy model. Our algorithm requires just 0.2 seconds for this example. Note that 58.1% of the triangles have a small angle (less than $\frac{\pi}{6}$).

a 2D Euclidean plane. This makes the restricted Voronoi diagram (RVD) [29] become a popular tool for computing the surface constrained Voronoi diagram, where the region dominated by a generator is the intersection between the surface and the corresponding 3D Voronoi cell. However, only when the local feature size (LFS) sampling condition is satisfied, the RVD can serve as the alternative of the GVD. And for a sparse collection of generators, RVD cannot be taken as an alternative of GVD any more. When the surface is highly curved or the sites are very spare, the resulting RVD may produce many flawed regions due to the inconsistency between straight-line distance and geodesic distance, as pointed out in [28, 24]. In this situation, one has to use exact or approximate geodesic distance instead, and infer the GVD based on the competition between geodesic wavefronts from different generators.

Liu et al. [14] pointed out that an exact GVD may consist of straight-line segments and hyperbolic curve segments on a given polygonal surface. Therefore, computing an exact GVD is very time-consuming yet unnecessary in real computer graphics scenarios. There are several research works on computing approximate GVDs. Xu et al. [27] used an exact geodesic algorithm and suggested keeping the windows enclosing each triangle and computing an additively weighted Voronoi diagram, which can obtain an accurate GVD and run relatively fast. Herholz et al. [8] suggested estimating the diffusion distances as approximate geodesic distances for each generator to cover its dominating region, followed by tracing GVD segments based on an assumption that the distance change given by a generator is linear in the mesh edge. It can be seen that most of the existing algorithms include two stages: 1) propagating geodesic wavefronts from generators simultaneously until every mesh vertex finds the nearest generator, and 2) competing between different geodesic wavefronts to predict the GVD structure.

However, both the two stages require a lot of computation. In this paper, we invent two techniques to speed up the computation. For the first stage, we adapt the fast marching method (FMM) [22] to drive the computation of geodesic distances, and allow the geodesic wavefronts, for each generator, to cover an as-small-as-possible range. For the second stage, we first compute the ordinary GVD segments and then predict the branching points, which is inspired by an interesting observation that the branching structure can be inferred by stretching the ordinary GVD segments until they converge. Figure 1 demonstrates the result of our method on Lucy model.

The proposed algorithm workflow distinguishes itself from the existing GVD algorithms. Firstly, it decouples the computation of the GVD from the usage of a specific geodesic algorithm. Secondly, as it is time consuming to deduce the structure around a branching point, we use a growth-based technique to stretch the traced ordinary GVD segments, rather than compute the GVD branching structures directly based on the geodesic distances. Last but not least, the GVD generally passes through only a small portion of triangles, which motivates us to adapt the FMM such that each vertex in the GVD band receives the "signal" from the first two nearest generators. Experimental results show that our algorithm runs on average of 3 times as fast as the state-of-the-art method while keeping a small accuracy difference.

2. Related Work

This section mainly reviews two kinds of related research works, i.e., geodesic distance computation and GVDs.

2.1. Geodesic Distance Computation

The problem of computing geodesic distances on a triangular mesh is often referred to as the discrete geodesic problem [16]. Methods for calculating discrete geodesics can be roughly divided into partial differential equation (PDE) methods [22, 7, 10, 25] and computational geometry methods [5, 23, 26, 1]. The former methods solve the Eikonal equation with boundary condition, while the latter methods propagate a discrete wavefront in a sweeping fashion. Generally speaking, PDE methods run fast but cannot achieve high accuracy, whereas propagation-based methods vary in their ability to maintain the balance between accuracy and speed. To our best knowledge, the

VTP algorithm [21] is the most efficient exact geodesic algorithm. We refer the readers to [6] for a comprehensive survey.

2.2. Voronoi Diagrams

Voronoi diagrams in Euclidean spaces have been extensively studied [4, 9]. However, Voronoi diagrams in non-Euclidean spaces are much different. There are some research works that construct differential forms on spheres [2, 17], hyperbolic spaces [19], regular parametric surfaces [11], and Riemannian manifolds [18, 3].

In the field of computer graphics, the typical representation of a 3D object is a polygonal triangle mesh. The computation of GVDs needs to overcome two challenges. The first is to quickly infer the geodesic distance, and the second is to trace how the GVD structure passes through the triangles. Existing GVD algorithms can be roughly divided into three groups. The first group is to trace the GVD by considering how a geodesic algorithm sweeps the surface. For example, Liu et al. [14] consider the propagation process of the MMP algorithm [16] and identify the locus where two windows from different generators compete. Later, Xu et al. [27] improved the computation process. The algorithm suggests keeping the windows enclosing each triangle and computing an additively weighted Voronoi diagram. For the GVD algorithms in this group, the computational cost consists of propagating wavefronts from generators simultaneously and inferring geodesic bisectors by competition between different generators, where the latter operation is the computational bottleneck, especially deducing the structure around a branching point. The second group is to trace the GVD structure simply from the multi-source diffusion field. For example, Herholz et al. [8] calculated heat diffusion field for each generator and computed approximate bisector under the linear field assumption. Algorithms in this group cannot produce accurate GVD results. The last uses RVD to produce GVDs. RVD works well for sufficiently dense generators, but is weak to deal with a sparse set of generators. For example, RVD may fail for a thin-plate structure. Localized RVD [28, 24], as an extension of RVD, aims at tackling this issue but it may produce significant errors due to the inconsistency between straight-line distance and geodesic distance.

In this paper, we consider the GVD problem assuming that there is a sparse set of generators. In this situation, one has to use geodesic distance to drive the computation of GVDs. In our implementation, we use the FMM [22, 10] to estimate geodesic distances.

3. Insight

The computation of GVDs. The GVD can be viewed as an extension of the traditional Voronoi diagram from Euclidean spaces to curved surfaces. It has to use geodesic distances to measure the proximity between points, resulting a partitioning scheme of the surface. A point q is said to be on the GVD if and only if q is equally distant to its two nearest generators. All such points define the GVD. The computation consists of two parts. The first part is to propagate distances from each generator outward and infer how a mesh vertex gets the geodesic distance from the nearest generator. The second part is to create competition among the generators such that all surface points with two equally-distant nearest generators can be identified.

Why the multi-source distance field doesn't work. A natural idea for keeping track of the distances from vertices to generators is to run a geodesic algorithm to compute the multi-source distance field where each generator servers as a source point. However, when the distance propagation terminates, each vertex can only get the first nearest generator, lacking the clue about the second nearest generator or even the third nearest generator. The GVD contains the two types of points:

- **Ordinary GVD points:** there are two generators equally distant to the GVD point, giving the smallest distance.
- **Branching GVD points:** there are at least three generators equally distant to the GVD point, giving the smallest distance.

As in a triangle f, each vertex maintains only its contributing generator, it does not suffice to infer how the contributing generators compete in f.

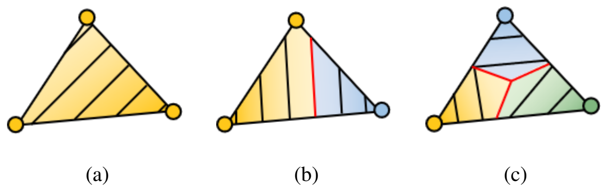


Figure 2. We can divide the triangles into three groups, depending on the number of contributing generators. (a) Trivial triangle. (b) Ordinary triangle. (c) Branching triangle.

Allowing two geodesic distance fields to overlap. For the triangles past through by the GVD, one has to keep track of the two or three nearest generators. For this purpose, we assume the generator s_1 provides three distance values to the triangle $f = \triangle ABC$, i.e.,

$$d(A; s_1), d(B; s_1), d(C; s_1).$$

Similarly, the generator s_2 also provides three distance values, i.e.,

$$d(A; s_2), d(B; s_2), d(C; s_2).$$

We say s_1 defeats s_2 if

$$d(A;s_1) < d(A;s_2), d(B;s_1) < d(B;s_2), d(C;s_1) < d(C;s_2).$$

Obviously, it is possible that none of s_1 and s_2 is the overcomer. Therefore, in a triangle-wise propagation style, it is likely that two or more generators survive at a triangle when the propagation process terminates. To this end, a triangle may receive one generator or two generators (three vertices share two contributing generators) or three generators (different vertices own different contributing generators). We thus divide the triangles into three groups, as Figure 2 shows:

- **Trivial triangles:** three vertices share just one contributing generator.
- **Ordinary triangles:** three vertices share two contributing generators.
- **Branching triangles:** different vertices own different contributing generators.

As long as the triangulation quality is not too bad, it is reasonable to assume that the ordinary triangles and the branching triangles accommodate the whole GVD structure. To better infer how a generator contributes to the distance field constrained in a triangle, we add edge midpoints as auxiliary points to record more distance values during the propagation process.

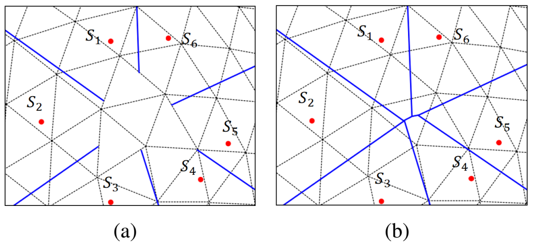


Figure 3. In 2D, the Voronoi diagram can grow from an incomplete configuration to a complete configuration. (a) The ordinary segments are known but the branching structures are missing. (b) The branching structures can be completely inferred from the incomplete ordinary segments.

Why the GVD has the growth ability. The GVD can be computed based on the multi-source wavefronts from different generators. Generally speaking, for an Ordinary-type triangle, the wavefronts come from two different directions, which is manageable. But for the Branching-type triangles, three or more wavefronts arrive at one triangle simultaneously, making the competition between generators more complicated and thus requiring tedious computation. We observe that the GVD edges are mostly straight and smooth, like the Voronoi edges in 2D. The rigidity of the GVD edges inspires us to predict the complicated GVD branching structures based on growth. As Figure 3 shows, the 2D Voronoi diagram can grow from an incomplete configuration to a complete configuration. In the following, we will explain the rationale behind from two aspects.

As Figure 3(a) shows, there are totally 6 generators:

$$s_i = (x_i, y_i), i = 1, 2, 3, 4, 5, 6,$$

thus giving 12 freedom degrees. Each symmetry relationship, given by one perpendicular bisector, eliminates 2 freedom degrees. Therefore, this implies that the 6 perpendicular bisectors (colored in blue) can encode the positions of the 6 generators, which further determine the complete Voronoi diagram. (It must be pointed out that if some coincidence occurs, there may be more freedom degrees remaining. For example, given four generators $(-1,-1), (-1,1), (1,-1), (1,1)$, one can move one of them arbitrarily but keep their Voronoi diagram unchanged.) The observation implies that the traced ordinary segments shown in Figure 3(a) are able to determine the missing structures.

We can study the problem from a different perspective. As Figure 3(a) shows, the 6 generators can be arranged in a circular order. Each pair of successive generators gives a perpendicular bisector, thus having 6 ordinary Voronoi edges pointing outward. Every branching point is determined by 3 generators and equally distant to them. Therefore, the missing part of the Voronoi diagram can be predicted according to the two principles: 1) repeatedly merge two successive ordinary Voronoi edges, and 2) accept the intersection as a Voronoi vertex if the intersection is equally distant to the 3 corresponding generators but more distant to the other generators.

4. Growth-based Algorithm

We assume that a set of generators $S = \{s_i\}_{i=1}^m$ are specified on a watertight manifold triangular mesh M. The task is to compute the polyline representation of the GVD, and report the connections between the GVD branching points.

Our algorithm pipeline is shown in Figure 4. The algorithm begins with computing the overlapping distance field using the adapted FMM, then constructs ordinary GVD bisectors constrained in those triangles with two or more contributing generators, and finally stretches the ordinary GVD bisectors to a complete GVD based on a growth technique. We will detail the important steps later.

4.1. Adapt The FMM for Producing Overlapping Distances

FMM. As a typical mark-and-sweep algorithm, the FMM utilizes a priority queue to propagate distance events from

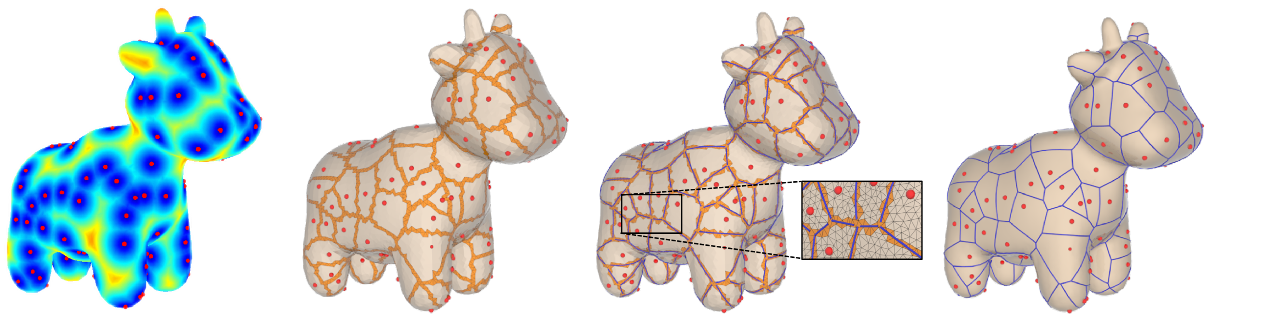

(a) (b) (c) (d)

Figure 4. Algorithm pipeline. (a) Compute the overlapping distance field using the adapted FMM, where the generators are colored in red. (b) Once the overlapping distance field is computed, those triangles with two or more contributing generators are found (colored in orange). (c) Construct ordinary GVD bisectors constrained in the orange triangles. (d) Stretch the ordinary GVD bisectors until the complete GVD is traced.

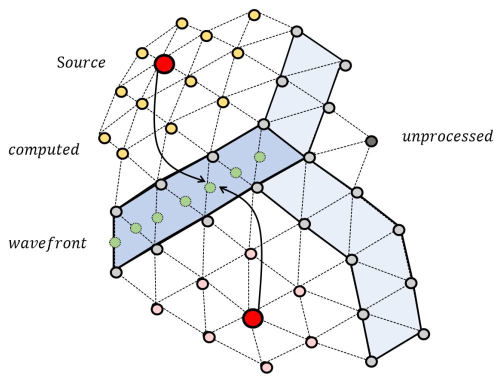


Figure 5. The FMM is a typical mark-and-sweep algorithm. During its execution, the wavefronts propagate outward until the distances of all the vertices cannot be reduced.

near to far and prioritizes events closest to the generators, as shown in Figure 5. During its execution, the wavefronts propagate outward until the distances of all the vertices cannot be reduced. It terminates when each vertex finds its nearest generator, producing a multi-source distance field. The FMM runs many times faster than the exact geodesic algorithms, and can predict geodesic distances very accurately especially when the triangles are dense and have a good quality. However, such a distance field is not informative enough. For the triangles that accommodate the GVD, it is hard to accurately predict how the GVD passes through the interior only by the three distances values kept at the triangle, which motivates us to adapt the FMM to satisfy the requirements of computing GVDs.

Adapt the FMM. We use two key techniques to improve the FMM. First, we propagate the wavefronts in a triangle-wise style, rather than in a vertex-wise style like that achieved in the traditional FMM. For this purpose, we maintain a generator table, for each triangle, to keep those generators that may provide shortest distances to at least one point of this triangle. Let $f = \triangle ABC$ be a triangle swept by the generator s. Then f's priority is measured by

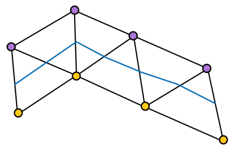
$$d(A; s) + d(B; s) + d(C; s).$$

The wavefronts transmitted by s are not allowed to cross f if s is totally defeated by another generator. Second, we take the midpoint of each mesh edge as an auxiliary point and keep a distance value for each contributing generator. Suppose that s is one of the contributing generators kept in the triangle f. Recall that the FMM infers a linear distance field for each triangle, when s propagates its distances to f, it is easy to evaluate the distance for an arbitrary point in f, naturally including the three edge midpoints. Obviously, the new FMM, adapted by the two techniques, can give richer information about how a generator contributes to the overall distance field. Particularly, the adapted FMM keeps more distance clues for those ordinary triangles and branching triangles.

Why the FMM suffices. Two reasons account for why we use the FMM, instead of those exact geodesic algorithms. On one hand, exact geodesic algorithms run slowly for large-sized models. On the other hand, GVD bisectors serve as the symmetry axis between a pair of adjacent generators. Therefore, even if we use a transformed distance form like $\mathbf{T}(d(\cdot, \cdot))$, the traced GVD remains unchanged, where $d(\cdot, \cdot)$ reports the exact geodesic distance and $\mathbf{T}(\cdot)$ defines a transform that increases monotonically.

4.2. Construct Ordinary GVD Bisectors

We accomplish the tracing of ordinary GVD bisectors in two stages, i.e., connecting a sequence of midpoints into a polyline to extract the rough shape of the ordinary GVD bisectors, and straightening the rough ordinary GVD bisectors by local unfolding.



Rough ordinary GVD bisectors. A triangle is defined to be an ordinary triangle if the three vertices own two different contributing generators. Let $f = \triangle ABC$ be an ordinary triangle. Suppose that the closest generator to A is s_1, while the closest generator to B, C is s_2. We then connect the midpoint of AB and the midpoint of AC into a straight-line segment, which is named a *rough* segment. All the rough segments form the rough representation of the ordinary part of the GVD. The inset figure shows an example of the rough representation of the midpoint-based ordinary part, where the vertices colored in yellow share one generator and the vertices colored in purple share the other generator.

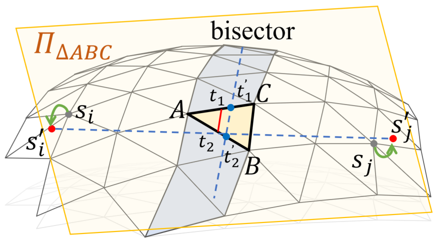


Figure 6. Refine ordinary GVD segments. Let $\Pi_{\triangle ABC}$ be the plane of $\triangle ABC$. s_i (resp. s_j) is unfolded onto $\Pi_{\triangle ABC}$ according to $d(t_1; s_i)$ and $d(t_2; s_i)$ (resp. $d(t_1; s_j)$ and $d(t_2; s_j)$), yielding an image point s_i' (resp. s_j'). The two images s_i' and s_j' can fine-tune the GVD segment to a better configuration.

Refine ordinary GVD bisectors. Suppose that $\triangle ABC$ is an ordinary triangle, as defined in Section 3. There are two generators s_i, s_j that give the shortest distances to A, B, C. Take Figure 6 as an example. The generator s_i gives the shortest distance to A while the generator s_j gives the shortest distances to B and C. According to the adapted FMM discussed in Section 4.1, the distances from s_i to A, B, C, as well as the three edge midpoints, must be kept at the same time. Similarly, the distances from s_j to the three edge midpoints are kept as well. The generator s_i can be unfolded onto the plane of $\triangle ABC$ based on $d(t_1; s_i)$ and $d(t_2; s_i)$, where t_1, t_2 are the edge midpoints. Let s_i', s_j' be respectively the unfolded image points. It is easy to infer a better configuration of the GVD segment.

Why the adapted FMM helps. Recall that the adapted FMM has two differences from the traditional FMM. On one hand, it propagates in a triangle-wise fashion, rather than a vertex-wise fashion. In Figure 6, s_i is the closest generator for the vertex A, and thus s_i must arrive at the entire triangle $\triangle ABC$, enabling the distance $d(t_1; s_i)$ and $d(t_2; s_i)$ available. Additionally the segment t_1t_2, in spite of not accurately aligning with the real GVD, is closer to the configuration of the GVD than AB, BC, and AC. Consequently, the segment t_1t_2 helps better predict the position of the image points s_i' and s_j'.

Consistency enforcement. Note that the operation of refining ordinary GVD segments works triangle by triangle. But it cannot guarantee the consistency across mesh edges for the mesh is not always a developable surface so that the expansion operation only provide an approximate result. Let f_1, f_2, sharing a common edge e, be two ordinary triangles. Each of them reports a GVD segment. However, the two ordinary segments, upon being refined, may not join at the same point on e, thus violating the consistency. Let $t^e_{f_1}, t^e_{f_2}$ be the two endpoints on e, given by f_1 and f_2 respectively. Therefore, we enforce the consistency by simply averaging the two endpoints $t^e_{f_1}$ and $t^e_{f_2}$.

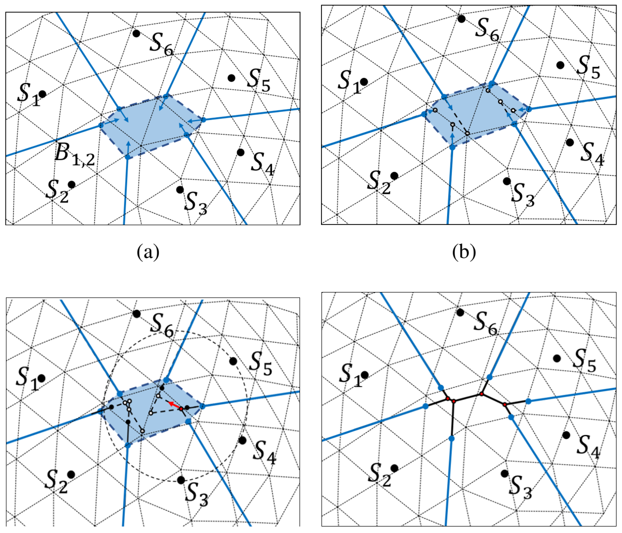
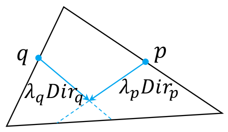


Figure 7. Growth Algorithm. (a) Only some ordinary GVD segments are known, leaving the GVD part around a branching point missing. (b) From the broken ends, we stretch the ordinary GVD segments. The white dots represent events currently in the priority queue. The situations are of two kinds, i.e., exiting from a triangle and intersecting inside a triangle, which can be observed in (c). (c) Prioritize the events with the least distance values. (d) When all the events are handled, the missing part is filled.

4.3. Stretch Ordinary GVD Bisectors

Two kinds of events. We will discuss the growth-based strategy for generating a complete GVD. The original intention is to predict the branching structures depending on the traced ordinary GVD segments, rather than by a lot of distance queries. Figure 7 shows the key steps of the growth-based algorithm for filling the missing part. In the 2D Euclidean plane, the growth process is conducted by repeated merging from the broken ends. For a polygonal mesh, however, one has to conduct the growth process from triangle to triangle. There are two situations to be considered, i.e., exiting from a triangle and intersecting inside a triangle. In implementation, we use a priority queue to maintain the two kinds of discrete events such that the event with the least distance is first handled. When all the events are handled, the missing part is filled.

Intersection computation. Suppose that the GVD enters f from the point p and the point q, with the entry directions being Dir_p and Dir_q respectively. The intersection point between them can be computed by



$$p + \lambda_p \text{Dir}_p = q + \lambda_q \text{Dir}_q,$$

from which λ_p and λ_q can be solved quickly. The coordinates of the intersection point can be thus computed. The growth direction of the intersection point will be predicated along with the distance predication by unfolding two corresponding generators to the plane of f. If the intersection is not inside f, then it is judged to be invalid and thus ignored. In addition, the exit point of a GVD segment can be similarly computed.

Rotating around an edge. Suppose that the segment pq is leaving the face f_1. It arrives at the point q on the bounding edge e, and will enter the plane of the next face f_2. Let $\mathbf{n}_1, \mathbf{n}_2$ be respectively the normal vectors of f_1 and f_2. The key spirit of unfolding f_1 and f_2 onto one plane is to rotate f_1 around e such that f_1's normal vector is consistent with f_2's normal vector. That is to say, the vector in Π_{f_1}

$$\mathbf{n}_1 \times \mathbf{e}$$

becomes

$$\mathbf{n}_2 \times \mathbf{e}$$

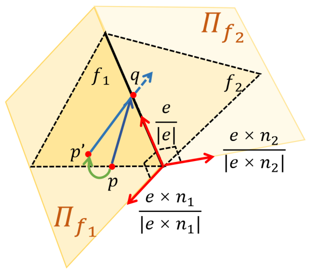
upon being unfolded. Therefore, we first decompose pq as follows:

$$pq = (pq \cdot \frac{\mathbf{e}}{\|\mathbf{e}\|})\frac{\mathbf{e}}{\|\mathbf{e}\|} + (pq \cdot \frac{\mathbf{n}_1 \times \mathbf{e}}{\|\mathbf{n}_1 \times \mathbf{e}\|})\frac{\mathbf{n}_1 \times \mathbf{e}}{\|\mathbf{n}_1 \times \mathbf{e}\|}.$$

We then can obtain $p'q$:

$$p'q = (pq \cdot \frac{\mathbf{e}}{\|\mathbf{e}\|})\frac{\mathbf{e}}{\|\mathbf{e}\|} + (pq \cdot \frac{\mathbf{n}_1 \times \mathbf{e}}{\|\mathbf{n}_1 \times \mathbf{e}\|})\frac{\mathbf{n}_2 \times \mathbf{e}}{\|\mathbf{n}_2 \times \mathbf{e}\|},$$

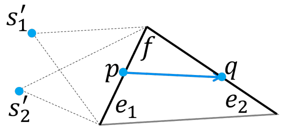
which is unfolded onto the plane of f_2.



Π_{f_2}
f_1
q
f_2
$\frac{e}{|e|}$
p'
$\frac{e \times n_2}{|e \times n_2|}$
p
$\frac{e \times n_1}{|e \times n_1|}$
Π_{f_1}

Figure 8. The segment pq is leaving the face f_1 and will enter the plane of the next face f_2.

Distance predication. For an ordinary segment in the face f, it has an entry edge e_1 and an exiting edge e_2. When it arrives at an edge e_2, yielding an intersection $q \in e_2$. Suppose that the two generators associated with the ordinary segment are s_1 and s_2. The two generators can be easily unfolded to the plane of f, resulting in two image points s_1' and s_2'. It is easy to compute the distance between s_1' and q, as well as the distance between s_2' and q. By taking the average value, we can estimate the distance at q contributed by s_1 and s_2.



Similarly, when two ordinary segments intersect each other in a triangle f, the intersection point is given by three generators s_1, s_2, s_3. We unfold $s_i, i = 1, 2, 3$, onto the plane of f around s_i's entry edge. The distance at the intersection point can be estimated by averaging the three distance values. Furthermore, the three image points can help predict the next growth direction from the intersection point.

4.4. Complexity Analysis

To estimate the time complexity, we first show an interesting observation. Let n_1 be the number of trivial triangles, n_2 be the number of ordinary triangles, and n_3 be the number of branching triangles. $n = n_1 + n_2 + n_3$ is the total number of triangles. When the base surface has dense triangulation, we observe $n_1 \gg n_2 \gg n_3$, which implies that the adapted FMM does not significantly increase the overhead. Therefore, the overall propagation cost can be bounded by $O(n \log n)$, the same with the total timing cost of the traditional FMM.

During the process of constructing the GVD, we need to trace the broken ends of the ordinary GVD segments, and the number of broken ends can be bounded by the total number of faces, i.e., $O(n)$. We also need to trace the GVD branching points whose number is $O(m)$, where m is the number of generators. The peak length of the priority queue for maintaining the order of events being processed is thus $O(n + m)$, with each push/pop operation being accomplished in $O(\log(n + m))$.

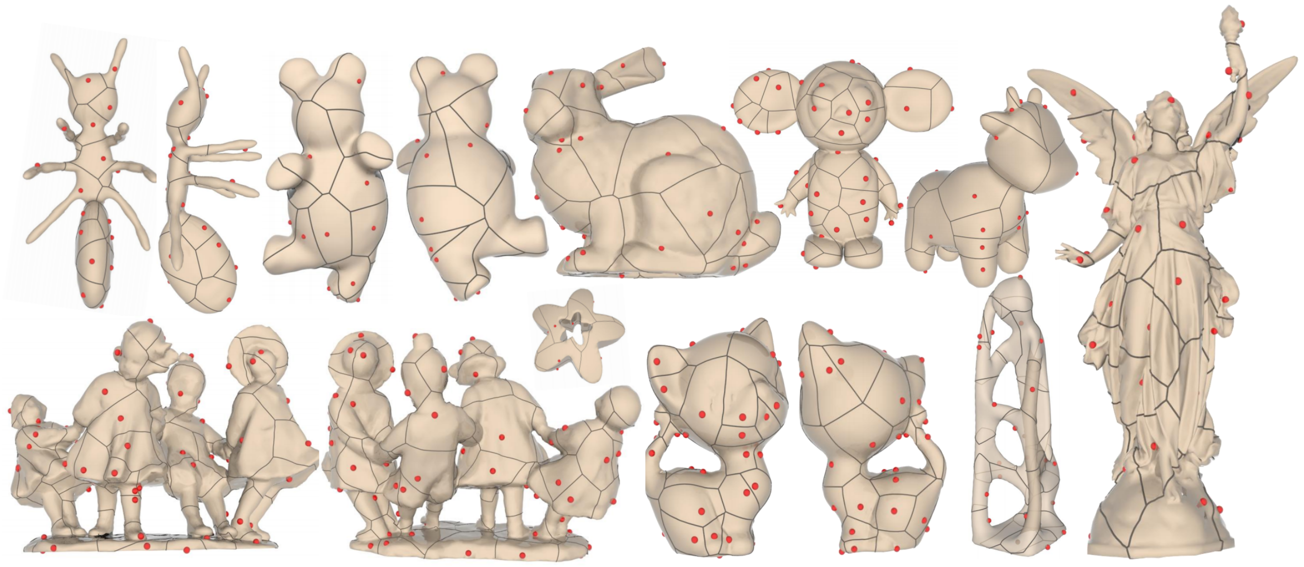


Figure 9. A gallery of GVD results computed by our algorithm.

Therefore, the overall time complexity can be bounded by $O((n+m)\log(n+m))$.

5. Evaluation

To evaluate the performance of the proposed algorithm, we implement it in C++. All the experiments were conducted on a PC with Intel(R) Core(TM) i5-8400 CPU and 8GB memory. As our algorithm is stronger in dealing with sparse generators, the number of generators in the experiments ranges from 10 to 500. (When there are enough generators, it is better to use the restricted Voroni diagram [29] instead.) Figure 9 is the gallery of the GVD results computed by our approach on a variety of models. It shows that our method can process not only simple models but also geometry/topology complicated complex models.

5.1. Run-time Performance

The whole GVD computation procedure includes two parts: (1) computing multi-source geodesic distance field and (2) predicting GVD bisectors on. We will come to evaluate our algorithm about how the timing cost varies with the mesh complexity and the number of generators.

Number of facets. We randomly select 50 vertices as generators and repeatedly increase the mesh resolution. Table 1 shows the timing statistics. The overhead of computing the geodesic distance field (GDF) increases with the number of faces. At the same time, the overhead of tracing the GVD increases as well.

F	GDF(ms)	Bisector(ms)	Total(ms)	Memory(KB)
10k	76.11	19.25	95.3	293.59
30k	366.2	187.6	553.7	464.87
60k	758.65	231.12	989.8	571.68
90k	1967.8	276.98	2244.8	703.70
120k	3893.7	1042.02	4935.7	799.31
150k	6701.08	627.39	7328.4	877.07

Table 1. Run-time performance statistics on the Kitten model with 50 generators but increasing mesh resolutions.

m	GDF(ms)	Bisector(ms)	Total(ms)	Memory(KB)
10	96.62	7.66	104.28	127.91
30	117.1	20.63	137.73	261.63
60	162.2	31.89	194.12	415.87
80	171.0	41.00	211.99	510.14
120	238.4	64.83	303.28	665.10
250	197.9	93.83	292.77	1023.87
500	330.2	164.83	495.155	1,766.63

Table 2. The run-time performance of our algorithm on the Spot Model (V=10k, F=20k). We keep the base unchanged while increasing the number of generators, denoted by m.

Number of generators. By keeping the base surface unchanged, we repeatedly increase the number of generators. For example, the number of generators on the Spot model increases from 10 to 500. Table 2 shows the timing statistics and the memory usage with the increasing of the number of generators. The main reason lies in that as the number of generators grows, so does the number of branching triangles, which raises timing consumption.

To summarize, the empirical observation is consistent with our theoretical time complexity $O(m+n)\log(m+n)$.

But we must point that when the generators are sufficiently many, the restricted Voroni diagram [29] is a better choice. When the generators are sparse, our algorithm has a bigger advantage.

5.2. Accuracy v.s. Mesh Quality

Our method is not sensitive to mesh quality. Although it performs on a poor-quality mesh, the resulting GVD is still favourable. To evaluate correctness of our result, we use stochastic error calculated on GVD bisectors. Let p be a point sitting on the GVD. It's known that p must be equally distant to the two nearest generators. Therefore, it is proper to use the following function to measure the total deviation of the traced GVD to the exact GVD:

$$\mathcal{E} = \int_{\mathrm{GVD}} E(p)\mathrm{d}p,$$

where $E(p)$ can be defined as follows. Let $S(p) = \{s_1, s_2, \cdots, s_k\}$ denote the generator set related to p. We evaluate the geodesic distance $d(p; s_i)$ for each $i = 1, 2, \cdots, k$. Then $E(p)$ is given by the maximum difference:

$$E(p) = \max_{i,j} |d(p; s_i) - d(p; s_j)|.$$

Besides, we use $Q(t)$ to measure the quality of the triangle t, where $Q(t) = \frac{6}{\sqrt{3}} \frac{|t|}{p(t)h(t)}$ [28] ($|t|$ is the area of t. $p(t)$ is the half-perimeter of t and $h(t)$ is the longest edge length of t). When $Q(t)$ approaches 1, the triangular mesh has the best quality (each triangle facet is regular triangle). To measure the quality of the model, we use a pair of indicators, i.e., Q_{avg} and Percent($< 30°$), where the latter denotes the percentage of triangles with minimum angles less than $30°$.

Model	Quality	Q_{avg}	Percent($< 30°$)	$\mathcal{E}$
Elephant	low	0.598	52.88	0.087
	medium	0.769	10.15	0.132
	high	0.859	0.455	0.076
Rocker	low	0.692	27.57	0.303
	medium	0.803	11.88	0.154
	high	0.914	0.00	0.222
Pegaso	low	0.638	39.43	0.164
	medium	0.758	11.62	0.126
	high	0.871	00.02	0.120

Table 3. Accuracy statistics for various meshing quality.

Table 3 gives the accuracy statistics of our method on polygonal surfaces with varying meshing quality. It can be seen that the accuracy does not significantly drop even if the input mesh has a poor quality. The reasons are two-fold. First, we also consider the distance values at the auxiliary points, which is more informative than the traditional FMM. Second, the growth based strategy aims at inferring the missing branching structure based on the traced ordinary segments, and thus insensitive to mesh quality.

5.3. Comparison with Xu et al. [27]

Xu et al. [27] propose a geodesic distance-based algorithm to construct GVDs. It can be viewed as the state-of-the-art in accurately computing a GVD. Similar to our method, their method consists of two stages: (1) geodesic distance field computation via MMP [16] and (2) trace the GVD bisectors where the geodesic wavefronts terminate. Next, we will compare it with our approach from the two aspects: (1) run-time performance and (2) accuracy.

Run-time performance comparison. To make the comparison fair, we maximize the efficiency of Xu et al.'s method by setting the parameter $c = 1e^9$. As shown in Table 4, our algorithm runs 2 to 7 times faster than their algorithm. This is due to the fact that our method does not heavily depend on the tedious computation of geodesic distances especially in the branching triangles, unlike their algorithm, which has to construct GVD bisectors by competing competition between windows given by different generators. Besides, the FMM runs many times faster than the exact MMP algorithm.

Accuracy comparison. Table 4 shows that the accuracy difference between the two methods is surprisingly small. There are several reasons for this. First, the exact GVD consists of hyperbolic segments but existing algorithms can only trace straight-line segment, which implies that even if the exact geodesic algorithm is used, the resulting GVD is still approximate. Second, the FMM is not so accurate as the MMP, but the bias is roughly related to the distance. As pointed out earlier, a monotonically increasing transformation of distance field leads to the same GVD. Last but not least, we keep more informative clues about the distance field in the adapted FMM, which suffices to trace an accurate GVD.

5.4. Comparison with Diffusion Diagram [8]

The diffusion diagram proposed by Herholz et al. [8] takes advantage of the heat diffusion approach [7] for approximating geodesic distances. They suggested a recursive subdivision-based method to calculate GVDs. Based on our tests, the diffusion diagram is competitive on computing GVDs. Compared with our algorithm, the GVD reported by the diffusion diagram is less straight; See Figure 10.

The diffusion diagram has some inherent disadvantages. First, it has to utilize the cot-weight matrix and the factorization operation, which implies that the diffusion diagram is weak in dealing with poor triangulation and dense triangulation. For example, when the number of vertices

			Spot	Bunny	Ant	Lucy	Bear	Handok	4-kids
Model (F, m)			(5856, 30)	(10000, 47)	(14482, 24)	(19984, 43)	(20444, 21)	(20582, 35)	(25014, 83)
Xu et al.	Runtime (ms)	GDF	31.15	77.98	125.51	203.11	234.05	234.10	235.15
		VD	15.60	47.02	78.50	79.10	125.65	93.01	109.15
		Total	46.76	125.00	204.01	282.21	359.70	327.10	344.30
	$\mathcal{E}$(GVD)		0.168	0.151	0.141	1.095	0.016	0.042	0.308
Ours	Runtime (ms)	GDF	10.32	55.28	67.24	155.67	104.04	118.71	153.45
		VD	6.42	14.20	8.31	19.92	14.16	18.58	37.21
		Total	16.74	69.48	75.55	175.59	118.2	137.29	190.66
	$\mathcal{E}$(GBGVD)		0.205	0.184	0.154	1.196	0.019	0.073	0.562

Table 4. Performance comparison with [27]. "GDF" indicates the timing cost for computing a geodesic distance field while "VD" indicates the timing cost for computing Voronoi diagrams.

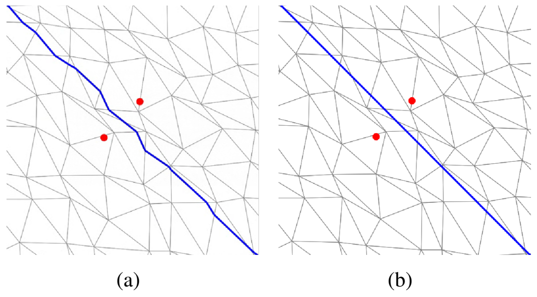


Figure 10. Comparison between Herholz et al. [8] and ours. Our GVD edges are straight but theirs are zig-zag. (a) Herholz et al.'s result. (b) Our result.

amounts to more than 10K, it is hard to compute the factorization of the large-sized dense cot-weight matrix. Second, each generator dominates a limited part of the surface, but it is hard for the diffusion diagram to accurately control the dominance region since it is not propagation based. Finally, the diffusion diagram needs to repeatedly split a triangle until the small-range distance change is linear. However, when the generators are very dense (e.g., many generators are located in one triangle), the algorithm may fail.

Model	(F, m)	$\mathcal{E}$(LRVD)	$\mathcal{E}$(GBGVD)	Ratio
sphere	(1280, 14)	0.024	0.016	1.50
spot	(5856, 30)	7.458	0.205	36.38
bunny	(10000, 47)	11.310	0.184	61.47
bear	(20444, 21)	13.001	0.019	684.26

Table 5. Comparison with the LRVD [28].

5.5. Comparison with RVD and LRVD [28]

The RVD is a popular choice when the generators are sufficiently dense. It can be viewed as the intersection of the 3D voronoi diagram and the input mesh surface (although its implementation depends on the Delaunay triangulation, rather than the 3D voronoi diagram itself).

Due to the inconsistency between Euclidean distances

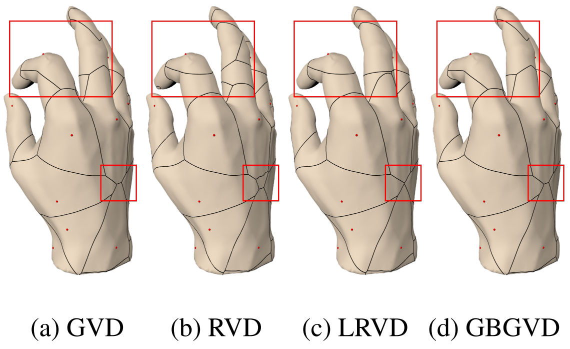


Figure 11. The RVD easily contains ownerless regions. Even if it is repaired and becomes the LRVD, the result is still conspicuously different from the exact GVD. In contrast to the LRVD, our result is more like the GVD.

and geodesic distances, the RVD may have various defects especially when the input shape contains thin plates or tubes. Yan et al. [28] proposed a sweeping-based approach to guarantee the "one generator, one cell" property, named *localized RVD* (LRVD), but all the boundary curves of a LRVD cell are still given by perpendicular bisectors in 3D. Therefore, the LRVD is fundamentally different from the GVD due to different mechanisms. It can be seen from Figure 11 that there is a major difference between the LRVD and the GVD in the highlighted area. Compared with the LRVD, our result is closer to the exact GVD. (We subdivide the surface into a dense version and run Xu et al.'s method [27] to simulate the exact GVD.)

6. Conclusions

In this paper, we present a simple yet efficient approach, named *GBGVD*, for computing the GVD on mesh surfaces. We propose two techniques to improve the efficiency and the accuracy. The first technique is the adapted FMM, which can obtain a more informative distance field with negligible additional overhead. The second technique is the growth-based GVD tracing algorithm. Rather than compute the branching structure purely using geodesic distances, we suggest using traced ordinary GVD segments to infer the missing branching structure. Our algorithm has a big advantage when the generators are sparse. However, when the generators are sufficiently dense, that is, each mesh triangle is a branching triangle, there is no ordinary triangle , and the ordinary segment cannot be inferred. At this time LRVD is a better choice than our approach in practice. We will further boost the performance of our approach in the future.

References

[1] Y. Y. Adikusuma, Z. Fang, and Y. He. Fast construction of discrete geodesic graphs. *ACM Trans. Graph.*, 39(2), mar 2020. 2

[2] J. M. Augenbaum and C. S. Peskin. On the construction of the voronoi mesh on a sphere. *Journal of Computational Physics*, 59(2):177–192, 1985. 3

[3] J.-D. Boissonnat, R. Dyer, and A. Ghosh. Constructing intrinsic delaunay triangulations of submanifolds. *arXiv preprint arXiv:1303.6493*, 2013. 3

[4] B. Boots, K. Sugihara, S. N. Chiu, and A. Okabe. Spatial tessellations: concepts and applications of voronoi diagrams. 2009. 3

[5] J. Chen and Y. Han. Shortest paths on a polyhedron. In *Acm Symposium on Computational Geometry*, pages 360–369, 1990. 2

[6] K. Crane, M. Livesu, E. Puppo, and Y. Qin. A survey of algorithms for geodesic paths and distances. *ArXiv*, abs/2007.10430, 2020. 3

[7] K. Crane, C. Weischedel, and M. Wardetzky. Geodesics in heat: A new approach to computing distance based on heat flow. *Acm Transactions on Graphics*, 32(5):13–15, 2013. 2, 9

[8] P. Herholz, F. Haase, and M. Alexa. Diffusion diagrams: Voronoi cells and centroids from diffusion. In *Computer Graphics Forum*, volume 36, pages 163–175. Wiley Online Library, 2017. 2, 3, 9, 10

[9] W. H. Hesselink and J. B. T. M. Roerdink. Euclidean skeletons of digital image and volume data in linear time by the integer medial axis transform. *IEEE Computer Society*, 2008. 3

[10] R. Kimmel and J. A. Sethian. Computing geodesic paths on manifolds. *Proceedings of the national academy of Sciences*, 95(15):8431–8435, 1998. 2, 3

[11] R. Kunze, F.-E. Wolter, and T. Rausch. Geodesic voronoi diagrams on parametric surfaces. In *Proceedings Computer Graphics International*, pages 230–237. IEEE, 1997. 3

[12] G. Leibon and D. Letscher. Delaunay triangulations and voronoi diagrams for riemannian manifolds. In *Sixteenth Symposium on Computational Geometry*, 2000. 1

[13] Y. Liu, W. Wang, B. Lévy, F. Sun, D.-M. Yan, L. Lu, and C. Yang. On centroidal voronoi tessellation—energy smoothness and fast computation. *ACM Transactions on Graphics (ToG)*, 28(4):1–17, 2009. 1

[14] Y.-J. Liu, Z. Chen, and K. Tang. Construction of isocontours, bisectors, and voronoi diagrams on triangulated surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1502–1517, 2010. 1, 2, 3

[15] L. Lu, B. Lévy, and W. Wang. Centroidal voronoi tessellation of line segments and graphs. In *Computer Graphics Forum*, volume 31, pages 775–784. Wiley Online Library, 2012. 1

[16] J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou. The discrete geodesic problem. *SIAM Journal on Computing*, 1987. 2, 3, 9

[17] H. S. Na, C. N. Lee, and O. Cheong. Voronoi diagrams on the sphere. *Computational Geometry*, 23(2):183–194, 2002. 3

[18] K. Onishi and J.-i. Itoh. Estimation of the necessary number of points in riemannian voronoi diagram. In *Proc. 15th Canad. Conf. Comput. Geom*, pages 19–24, 2003. 3

[19] K. Onishi and N. Takayama. Construction of voronoi diagram on the upper half-plane. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 79(4):533–539, 1996. 3

[20] J. Peethambaran and R. Muthuganapathy. Reconstruction of water-tight surfaces through delaunay sculpting. 58(C):62–72, jan 2015. 1

[21] Y. Qin, H. Yu, and J. Zhang. Fast and memory-efficient voronoi diagram construction on triangle meshes. In *Computer Graphics Forum*, volume 36, pages 93–104. Wiley Online Library, 2017. 3

[22] J. A. Sethian and A. Vladimirsky. Fast methods for the eikonal and related hamilton–jacobi equations on unstructured meshes. *Proceedings of the National Academy of Sciences*, 97(11):5699–5703, 2000. 2, 3

[23] V. Surazhsky, T. Surazhsky, D. Kirsanov, S. J. Gortler, and H. Hoppe. Fast exact and approximate geodesics on meshes. *ACM Transactions on Graphics*, 24(3):553–560, 2005. 2

[24] P. Wang, S. Xin, C. Tu, D. Yan, Y. Zhou, and C. Zhang. Robustly computing restricted voronoi diagrams (rvd) on thin-plate models. *Computer Aided Geometric Design*, 79:101848, 2020. 2, 3

[25] S. Q. Xin, D. T. P. Quynh, X. Ying, and Y. He. A global algorithm to compute defect-tolerant geodesic distance. In *Siggraph Asia Technical Briefs*, pages 1–4, 2012. 2

[26] S. Q. Xin and G. J. Wang. Improving chen and han's algorithm on the discrete geodesic problem. *Acm Transactions on Graphics*, 28(4), 2009. 2

[27] C. Xu, Y.-J. Liu, Q. Sun, J. Li, and Y. He. Polyline-sourced geodesic voronoi diagrams on triangle meshes. In *Computer Graphics Forum*, volume 33, pages 161–170. Wiley Online Library, 2014. 2, 3, 9, 10

[28] D.-M. Yan, G. Bao, X. Zhang, and P. Wonka. Low-resolution remeshing using the localized restricted voronoi diagram. *IEEE transactions on visualization and computer graphics*, 20(10):1418–1427, 2014. 2, 3, 9, 10

[29] D.-M. Yan, B. Lévy, Y. Liu, F. Sun, and W. Wang. Isotropic remeshing with fast and exact computation of restricted voronoi diagram. In *Computer graphics forum*, volume 28, pages 1445–1454. Wiley Online Library, 2009. 2, 8, 9