[Appendix]

The farthest-point geodesic Voronoi diagram of points on the boundary of a simple polygon*

Eunjin Oh[1], Luis Barba[2,3], and Hee-Kap Ahn[1]

1 **Department of Computer Science and Engineering, POSTECH**
77 Cheongam-Ro, Nam-Gu, Pohang, Gyeongbuk, Korea
{jin9082, heekap}@postech.ac.kr
2 **Département d'Informatique, Université Libre de Bruxelles**
Brussels, Belgium
lbarbafl@ulb.ac.be
3 **School of Computer Science, Carleton University**
Ottawa, Canada

Abstract

Given a set of sites (points) in a simple polygon, the farthest-point geodesic Voronoi diagram partitions the polygon into cells, at most one cell per site, such that every point in a cell has the same farthest site with respect to the geodesic metric. We present an $O((n+m)\log\log n)$-time algorithm to compute the farthest-point geodesic Voronoi diagram for m sites lying on the boundary of a simple n-gon.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases Geodesic distance, simple polygons, farthest-point Voronoi diagram

Digital Object Identifier 10.4230/LIPIcs.xxx.yyy.p

1 Introduction

Let P be a simple polygon with n vertices. Given two points x and y in P, the *geodesic path* $\pi(x,y)$ is the shortest path contained in P connecting x with y. Note that if the straight-line segment connecting x with y is contained in P, then $\pi(x,y)$ is a straight-line segment. Otherwise, $\pi(x,y)$ is a polygonal chain whose vertices (other than its endpoints) are reflex vertices of P. We refer the reader to [10] for more information on geodesic paths.

The *geodesic distance* between x and y, denoted by $d(x,y)$, is the sum of the Euclidean lengths of each segment in $\pi(x,y)$. Throughout this paper, when referring to the distance between two points in P, we mean the geodesic distance between them. To ease the description, we assume that each vertex of P has a unique farthest neighbor. This *general position* condition was also assumed by Aronov et al. [3] and Ahn et al. [2] and can be obtained by applying a slight perturbation to the positions of the vertices [7].

Let S be a set of m *sites* (points) contained in P. Given a point $x \in P$, a (geodesic) *S-farthest neighbor* of x, is a site $\mathrm{N}(P,S,x)$ (or simply $\mathrm{N}(x)$) of S that maximizes the geodesic distance to x. Let $F_S : P \to \mathbb{R}$ be the function that maps each point $x \in P$ to the distance to a S-farthest neighbor of x (i.e., $F_S(x) = d(x, \mathrm{N}(x))$). A point x in P that minimizes $F_S(x)$ is called the *geodesic center* of S (in P).

* This work was supported by the NRF grant 2011-0030044 (SRC-GAIA) funded by the government of Korea.

Conference title on which this volume is based on.
Editors: Billy Editor and Bill Editors; pp. 1–28

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

We can decompose P into *Voronoi cells* such that for each $s \in S$, $\mathsf{Cell}(s)$ is the set of points $x \in P$ such that $d(x, s)$ is strictly larger than $d(x, s')$ for any other site s' of S (some cells might be empty). The set $\text{int}(P) \setminus \cup_{s \in S} \mathsf{Cell}(s)$ defines the (farthest) *Voronoi tree* of S with root at the geodesic center of S and leaves on the boundary of P. Each edge of this diagram consists of a sequence of straight-lines and hyperbolic arcs [3].

The Voronoi tree together with the set of Voronoi cells defines the *farthest-point geodesic Voronoi diagram* of S (in P), denoted by $\mathsf{FVD}[S]$ (or simply FVD if S is clear from context). Thus, we indistinctively refer to FVD as a tree or as a set of Voronoi cells.

There are many similarities between the Euclidean farthest-point Voronoi diagram and the farthest-point geodesic Voronoi diagram (see [3] for further references). In the Euclidean case, a site has a nonempty Voronoi cell if and only if it is extreme, i.e., it lies on the boundary of the convex hull of the set of sites. Moreover, the clockwise sequence of Voronoi cells (at infinity) is the same as the clockwise sequence of sites along the boundary of the convex hull. With these properties, the Euclidean Voronoi diagram can be computed in linear time if the convex hull of the sites is known [1].

In the geodesic case, a site with nonempty Voronoi cell lies on the boundary of the relative convex hull. The clockwise order of the Voronoi cells along the boundary of P is a subsequence of the clockwise order of sites along the boundary of the relative convex hull of the sites. However, the cell of an extreme site may be empty, roughly because the polygon is not large enough for the cell to appear. In addition, the complexity of the bisector between two sites can be linear to the complexity of the polygon.

Previous work. Since the early 1980s many classical geometric problems have been studied in the geodesic setting. The problem of computing the geodesic diameter of the vertices of a simple n-gon P (and its counterpart, the geodesic center) received a lot of attention from the computational geometry community. Chazelle [6] gave the first algorithm for computing the geodesic diameter. This algorithm runs in $O(n^2)$ time using linear space. Suri [13] reduced the complexity to $O(n \log n)$-time without increasing the space complexity. Finally, Hershberger and Suri [9] presented a fast matrix search technique, one application of which is a linear-time algorithm for computing the diameter of P. A key step in this process is the computation of the farthest neighbor of each vertex in P.

The first algorithm for computing the geodesic center was given by Asano and Toussaint [4], and runs in $O(n^4 \log n)$-time. This algorithm computes a super set of the vertices of $\mathsf{FVD}[V]$, where V is the set of vertices of P. In 1989, Pollack et al. [12] improved the running time to $O(n \log n)$ time. In a recent paper, Ahn et al. [2] settled the complexity of this problem by presenting a $\Theta(n)$-time algorithm to compute the geodesic center of the vertices of a simple n-gon.

Since the geodesic center and diameter can both be computed from $\mathsf{FVD}[V]$ in linear time, the problem of computing farthest-point geodesic Voronoi diagrams is a strict generalization. For a set S of m points in P, Aronov et al. [3] presented an algorithm to compute $\mathsf{FVD}[S]$ in $O((n+m)\log(n+m))$ time. While a trivial lower bound of $\Omega(n + m \log m)$ is known for this general problem, there has been no progress closing this gap. In other words, it is not known whether or not the dependence on n, the complexity of P, is linear in the running time. In fact, this problem was explicitly posed by Mitchell [10, Chapter 27] in the Handbook of Computational Geometry.

Our result. In this paper, we present an $O((n+m)\log\log n)$-time algorithm to compute FVD of m points on the boundary of a simple n-gon. This is the first improvement on the

computation of geodesic farthest-point Voronoi diagrams since 1993 [3]. While we consider only sites lying on the boundary of the polygon, our result suggests that the complexity of the polygon has only a close-to-linear dependence in the computation of Voronoi diagrams. We believe our results could be used as a stepping stone to solve the question posed by Mitchell [10, Chapter 27].

Outline. The algorithm consists of three phases. First, we compute the farthest-point geodesic Voronoi diagram restricted to the boundary of the polygon. Then we recursively decompose the interior of the polygon into smaller (non-Voronoi) cells until the complexity of each of them becomes constant. Finally, we explicitly compute the farthest-point geodesic Voronoi diagram in each of the cells and merge them to complete the description of the Voronoi diagram.

In order to compute the Voronoi diagram of S, we start by computing the restriction of $\mathsf{FVD}[S]$ to the boundary of P in linear time. A similar approach was followed by Aronov et al. [3]. However, their algorithm spends $\Theta(n \log n)$ time and uses completely different techniques. The main tool used to speed up the algorithm is the matrix search technique introduced by Hershberger and Suri [9] which provides a "partial" description of $\mathsf{FVD}[S] \cap \partial P$, (i.e., the restriction of $\mathsf{FVD}[S]$ to the vertices of P.) To extend it to the entire boundary of P, we borrow some tools used by Ahn et al. [2]. This reduces the problem to the computation of upper envelopes of distance functions which can be completed in linear time.

Once $\mathsf{FVD}[S]$ restricted to ∂P is computed, we recursively split our polygon into cells. To split a cell whose boundary consists of k geodesic paths, we construct a closed polygonal path that visits roughly $\sqrt{k}$ endpoints of the k geodesic paths. Intuitively, To choose these endpoints, we start at the endpoint of a geodesic path bounding the cell. Then, we walk along the boundary and stop at another endpoint after skipping $\sqrt{k}$ of them, choose this endpoint and repeat. Between any two consecutive chosen endpoints we consider the geodesic path connecting them. The union of all these geodesic paths can be computed in time linear to the complexity of the cell [11] and produces smaller simple polygons. By recursively repeating this procedure on each resulting cell, we guarantee that after $O(\log \log n)$ rounds the boundary of each cell consists of a constant number of geodesic paths. In particular, we guarantee that each cell is either a pseudo-triangle, a quadrilateral or a simple polygon enclosed by a convex chain and a concave chain which we call a *lune-cell*.

While decomposing the polygon, we also compute the farthest-point geodesic Voronoi diagram of S restricted to the boundary of each cell. Each round can be completed in linear time which leads to an overall running time of $O((n+m)\log \log n)$.

Finally, we compute the farthest-point geodesic Voronoi diagram restricted to each cell in time proportional to the complexity of the cell and the diagram inside the cell using the algorithm in [5].

2 Decomposing the boundary

Given a set A of points, let ∂A and $\mathrm{int}(A)$ denote the boundary and the interior of A, respectively. Let P be a simple n-gon and S be a set of m sites (points) contained in ∂P. Throughout this paper, we will make the assumption that S is the set of all vertices of P. This assumption is general enough as we show how to extend the result to the case when S is an arbitrary set of sites contained on the boundary of P in Section 6.

The following result was used by Ahn et al. [2] and is based on the matrix search technique developed by Hershberger and Suri [9].

▶ **Lemma 1** (Result from [9]). *We can compute the S-farthest neighbor of each vertex of P in $O(n)$ time.*

Using Lemma 1, we mark the vertices of P that are S-farthest neighbors of at least one vertex of P. Let M denote the set of marked vertices of P (clearly this set can be computed in $O(n)$ time after applying Lemma 1). In other words, M contains all vertices of P whose Voronoi region contains at least one vertex of P.

For a marked vertex w of P, the vertices of P whose farthest neighbor is w appear contiguously along ∂P [3]. That is, given an edge uv such that $\textsc{n}(u) = \textsc{n}(v)$, we know that $\textsc{n}(x) = \textsc{n}(u) = \textsc{n}(v)$ for each point $x \in uv$. Therefore, after computing all these farthest neighbors, we effectively split ∂P into subchains, each associated with a different vertex of M (see [2] further for the first use of this technique).

Given two points x and y on ∂P, let $C[x, y]$ denote the portion of ∂P from x to y in clockwise order. We say that three (nonempty) disjoint sets A_1, A_2 and A_3 contained in ∂P are in *clockwise order* if $A_2 \subset C[a, c]$ for any $a \in A_1$ and any $c \in A_3$. (To ease notation, we say that three points $x, y, z \in \partial P$ are in clockwise order if $\{x\}, \{y\}$ and $\{z\}$ are in clockwise order).

For a polygonal chain A with endpoints p and q, we say A is a *transition chain* if $\textsc{n}(p) \neq \textsc{n}(q)$ and neither $\textsc{n}(p)$ nor $\textsc{n}(q)$ are interior vertices of A. In particular, if an edge ab of P is a transition chain, we say that it is a *transition edge*.

▶ **Lemma 2** ([3, Corollary 2.7.4]). *The order of sites with nonempty Voronoi cells along ∂P is the same as the order of Voronoi cells along ∂P.*

Let ab be a transition edge of P such that b is the clockwise neighbor of a along ∂P. Recall that we have computed $\textsc{n}(a)$ and $\textsc{n}(b)$ in the previous step and note that $a, b, \textsc{n}(a), \textsc{n}(b)$ are in clockwise order. Let v be a vertex of P such that $\textsc{n}(a), v, \textsc{n}(b)$ are in clockwise order. By Lemma 2, if there is a point x on ∂P whose farthest neighbor is v, then x must lie on ab. In other words, the Voronoi cell $\mathsf{Cell}(v)$ restricted to ∂P is contained in ab and hence, there is no vertex u of P such that $\textsc{n}(u) = v$.

Since we know which vertex is the farthest neighbor of each non-transition edge of P, to complete the description of FVD restricted to ∂P it suffices to compute FVD restricted to transition edges. To this end, we need some tools introduced in the following sections.

2.1 The apexed triangles

An *apexed triangle* $\triangle = (a, b, c)$ with *apex* $\textsc{a}(\triangle) = a$ is a triangle contained in P with an associated distance function $g_\triangle(x)$ such that (1) $\textsc{a}(\triangle)$ is a vertex of P, (2) there is an edge of ∂P containing both b and c, and (3) there is a vertex $\textsc{d}(\triangle)$ of P, called the *definer* of $\triangle$, such that

$$g_\triangle(x) = \begin{cases} \|x - \textsc{a}(\triangle)\| + d(\textsc{a}(\triangle), \textsc{d}(\triangle)) = d(x, \textsc{d}(\triangle)) & \text{if } x \in \triangle \\ -\infty & \text{if } x \notin \triangle, \end{cases}$$

where $\|x - y\|$ denote the Euclidean distance between x and y.

Intuitively, $\triangle$ bounds a constant complexity region where the geodesic distance function from $\textsc{d}(\triangle)$ can be obtained by looking only at the distance from $\textsc{a}(\triangle)$. We call the side of an apexed triangle $\triangle$ opposite to the apex the *bottom side* of $\triangle$. Note that the bottom side of $\triangle$ is contained in an edge of P.

The concept of the apexed triangle was introduced by Ahn et al. [2] and was key to their linear-time algorithm to compute the geodesic center. After computing the farthest

S-neighbor of each vertex, they show how to compute a linear number of apexed triangles in linear time with the following property: for each point $p \in P$, there exists an apexed triangle $\triangle$ such that $p \in \triangle$ and $\textsc{d}(\triangle) = \textsc{n}(p)$. By the definition of the apexed triangle, we have $d(p, \textsc{n}(p)) = g_{\triangle}(p)$. In other words, the distance from each point of P to its farthest neighbor is encoded in one of the distance functions associated with these apexed triangles. To summarize the results presented by Ahn et al. [2], we need some definitions. Given a chain C contained in ∂P with endpoints u and v, the *funnel* of a site s to C, denoted by $\gamma_s(C)$, is the weakly simple polygon contained in P bounded by C, $\pi(u, s)$ and $\pi(s, v)$.

▶ **Lemma 3** (Summary of [2]). *Given a simple n-gon P with vertex set S, we can compute a set of $O(n)$ apexed triangles in $O(n)$ time with the property that for any site $s \in S$, the union of each apexed triangle with definer s is a funnel γ_s such that* $\mathsf{Cell}(s) \subset \gamma_s$.

While Lemma 3 is not explicitly stated by Ahn et al. [2], a closer look at the proofs of Lemmas 5.2 and 5.3, and Corollaries 6.1 and 6.2 reveals that indeed, all the properties stated above hold. In other words, Lemma 3 states that for each site s of S, the set of apexed triangles with definer s forms a connected component. In particular, the union of their bottom sides is a connected chain along ∂P. Moreover, these apexed triangles are interior disjoint.

2.2 The refined farthest-point geodesic Voronoi diagram

We consider a refined version of FVD which we call the *refined farthest-point geodesic Voronoi diagram* defined as follows: for each site $s \in S$, the Voronoi cell $\mathsf{Cell}(s)$ of FVD is subdivided by the apexed triangles with definer s. That is, for each apexed triangle $\triangle$ with definer s, we define a *refined cell* $\mathsf{rCell}(\triangle) = \text{int}(\triangle) \cap \mathsf{Cell}(s)$. Since any two apexed triangles $\triangle_1$ and $\triangle_2$ with the same definer are interior disjoint, we know that $\mathsf{rCell}(\triangle_1)$ and $\mathsf{rCell}(\triangle_2)$ are also interior disjoint. We denote the set $\text{int}(P) \setminus \cup_{\triangle} \mathsf{rCell}(\triangle)$ by rFVD. Then, rFVD forms a tree consisting of arcs and vertices. Notice that each arc of rFVD is a connected subset of either the bisector of two sites or a side of an apexed triangle. Since the number of the apexed triangles is $O(n)$, the complexity of rFVD is still linear.

▶ **Lemma 4.** *Let x be a point in $\mathsf{rCell}(\triangle)$ for an apexed triangle $\triangle$. Then the line segment connecting x and y is contained in $\mathsf{rCell}(\triangle)$, where y is the point on the bottom side of $\triangle$ hit by the ray from $\textsc{a}(\triangle)$ towards x. Moreover, $\mathsf{rCell}(\triangle)$ is connected.*

Proof. Let p be a point on the line segment connecting x and y. We have $d(\textsc{d}(\triangle), p) = d(\textsc{d}(\triangle), x) + d(x, p)$. On the other hand, $d(s, p) \leq d(s, x) + d(x, p)$ by the triangle inequality for any site s. Since $d(s, x) < d(\textsc{d}(\triangle), x)$ for any site s other than $\textsc{d}(\triangle)$, we have $d(s, p) < d(\textsc{d}(\triangle), p)$, which implies that p lies in $\mathsf{rCell}(\triangle)$. This implies that $\mathsf{rCell}(\triangle)$ is connected since the bottom side of $\triangle$ is contained in $\mathsf{rCell}(\triangle)$. ◀

3 Computing the farthest-point geodesic Voronoi diagram restricted to the boundary of the polygon

Using the algorithms in [2], we compute all apexed triangles satisfying the condition in Lemma 3 in $O(n)$ time. Recall that the apexed triangles which have the same definer are interior disjoint and have their bottom sides on ∂P whose union forms a connected component chain along ∂P. Moreover, their union is a funnel by Lemma 3. Thus, the apexed triangles with the same definer can be sorted along ∂P.

▶ **Lemma 5.** *Let s be a site in S and let $\tau_s \neq \emptyset$ be the set of all apexed triangles with definer s. We can sort the apexed triangles in τ_s along ∂P with respect to their bottom sides in $O(|\tau_s|)$ time.*

Proof. Recall that the bottom side of an apexed triangle is contained in ∂P, and the other two sides are chords of P (possibly flush with ∂P). Assume that these chords are oriented from its apex to its bottom side. Using a hash-table storing the chords of the apexed triangles in τ_s, we can link each of these chords to its neighboring triangles (and distinguish between left and right neighbors). In this way, we can retrieve a linked list with all the triangles in τ_s in sorted order along ∂P. ◀

3.1 Computing rFVD restricted to a transition edge

Let uv be a transition edge of P. Without loss of generality, we assume that uv is horizontal and u lies to the left of v. Recall that if there is a site s with $\mathsf{Cell}(s) \cap uv \neq \phi$, then s lies in $C[\textsc{n}(v), \textsc{n}(u)]$. Thus, to compute $\mathsf{rFVD} \cap uv$, it is sufficient to consider the apexed triangles with definers in $C[\textsc{n}(v), \textsc{n}(u)]$. Let A be the set of apexed triangles with definers in $C[\textsc{n}(v), \textsc{n}(u)]$.

In this section, we give a procedure to compute $\mathsf{rFVD} \cap uv$ in $O(|A|)$ time using the sorted lists of the apexed triangles with definers in $C[\textsc{n}(v), \textsc{n}(u)]$. Once it is done for all transition edges, we have the refined farthest-point geodesic Voronoi diagram restricted to ∂P. Let $s_1 = \textsc{n}(u), s_2, \ldots, s_\ell = \textsc{n}(v)$ be the sites lying on $C[\textsc{n}(v), \textsc{n}(u)]$ in counterclockwise order along ∂P.

3.1.1 An upper envelope and rFVD

Consider any t functions $f_1, \ldots, f_t$ with $f_j : D \to \mathbb{R} \cup \{-\infty\}$ for $1 \leq j \leq t$, where D is any point set. We define the *upper envelope* of $f_1, \ldots, f_t$ as the function $f : D \to \mathbb{R} \cup \{-\infty\}$ such that $f(x) = \max_{1 \leq j \leq t} f_j(x)$. Moreover, we say that a function f_j *appears* on the upper envelope if $f_j(x) = f(x) \in \mathbb{R}$ at some point x.

Recall that a site s has the set of apexed triangles sorted along uv with respect to their bottom sides. Moreover, each such apexed triangle $\triangle$ has a distance function $g_\triangle$ such that $g_\triangle(x) = -\infty$ for a point $x \notin \triangle$ and $g_\triangle(x) = d(\textsc{d}(\triangle), x)$ for a point $x \in \triangle$. In this subsection, we restrict the domain of the distance functions to uv. By definition, the upper envelope of $g_\triangle$ for all apexed triangles $\triangle \in A$ on uv coincides with $\mathsf{rFVD} \cap uv$ in its projection on uv. We consider the sites one by one in order and compute the upper envelope of $g_\triangle$ for all apexed triangles $\triangle \in A$ on uv as follows.

While the upper envelope of $g_\triangle$ for all apexed triangles $\triangle \in A$ is continuous on uv, the upper envelope of $g_{\triangle'}$ of all apexed triangles $\triangle'$ with definers from s_1 up to s_k on uv (we simply say the upper envelope for sites from s_1 to s_k) might be discontinuous at some point on uv for $1 \leq k < \ell$. Let w be the leftmost point where the upper envelope for sites from s_1 to s_k is discontinuous. Then we define $U(s_k)$ as the function such that $U(s_k)(x)$ is the value of the upper envelope for sites from s_1 to s_k at x for a point x lying to the left of w, and $U(s_k)(x) = -\infty$ for a point x lying to the right of w. By definition, $U(\textsc{n}(v))$ is the upper envelope of the distance functions of all apexed triangles in A. Note that $\mathsf{rCell}(\triangle) \cap uv = \phi$ for some apexed triangle $\triangle \in A$. Thus the distance function of an apexed triangle might not appear on $U(s_k)$ on uv. Let $\tau_U(s_k)$ be the list of the apexed triangles whose distance functions appear on $U(s_k)$ sorted along uv from u with respect to their bottom sides. Note that if $\textsc{d}(\triangle_i) \neq \textsc{d}(\triangle_{i+1})$, the bisector of $\textsc{d}(\triangle_i)$ and $\textsc{d}(\triangle_{i+1})$ crosses the intersection of the bottom sides of $\triangle_i$ and $\triangle_{i+1}$ for two consecutive apexed triangles $\triangle_i$ and $\triangle_{i+1}$ of $\tau_U(s_k)$.

3.1.2 A procedure for computing $U(s_\ell)$

Suppose that we have already computed $U(s_{k-1})$ and $\tau_U(s_{k-1})$ for some index $2 \leq k \leq \ell$. We show how to compute $U(s_k)$ and $\tau_U(s_k)$ from $U(s_{k-1})$ and $\tau_U(s_{k-1})$ in the following. We use two auxiliary lists U' and τ'_U which are initially set to $U(s_{k-1})$ and $\tau_U(s_{k-1})$. We update U' and τ'_U until they finally become $U(s_k)$ and $\tau_U(s_k)$, respectively.

Let τ_k be the list of the apexed triangles with definer s_k sorted along ∂P with respect to their bottom sides. For any apexed triangle $\triangle$, we denote the list of the apexed triangles in τ_k overlapping with $\triangle$ in their bottom sides by $\tau_O(\triangle)$. Also, we denote the lists of the apexed triangles in $\tau_k \setminus \tau_O(\triangle)$ lying left to $\triangle$ and lying right to $\triangle$ with respect to their bottom sides by $\tau_L(\triangle)$ and $\tau_R(\triangle)$, respectively.

Let $\triangle_a$ denote the last element (the rightmost apexed triangle) of τ'_U. With respect to $\triangle_a$, we partition τ_k into three disjoint sublists $\tau_L(\triangle_a)$, $\tau_O(\triangle_a)$ and $\tau_R(\triangle_a)$. We can compute these sets in $O(|\tau_k|)$ time.

Case 1 : Some apexed triangles in τ_k overlap with $\triangle_a$. If $\tau_O(\triangle_a) \neq \phi$, let $\triangle$ be the leftmost apexed triangle in $\tau_O(\triangle_a)$. We compare the distance functions $g_\triangle$ and $g_{\triangle_a}$ on $\triangle_a \cap \triangle \cap uv$. That is, we compare $d(x, s_k)$ and $d(x, \mathrm{D}(\triangle_a))$ for $x \in \triangle_a \cap \triangle \cap uv$.

(1) If there is a point on $\triangle_a \cap \triangle \cap uv$ that is equidistant from s_k and $\mathrm{D}(\triangle_a)$, $g_\triangle$ appears on $U(s_k)$. Moreover, the distance functions of the apexed triangles in $\tau_R(\triangle)$ also appear on $U(s_k)$, and no apexed triangle in $\tau_L(\triangle)$ appears on $U(s_k)$ by Lemma 2. Thus we append $\triangle$ and the apexed triangles in $\tau_R(\triangle)$ at the end of τ'_U. We also update U' accordingly. Then, τ'_U and U' are $\tau_U(s_k)$ and $U(s_k)$, respectively.

(2) If $d(x, \mathrm{D}(\triangle_a)) \geq d(x, s_k)$ for all points $x \in \triangle_a \cap \triangle \cap uv$, then $\triangle$ and its distance function do not appear on $\tau_U(s_k)$ and $U(s_k)$, respectively, by Lemma 2. Thus we do nothing and scan the apexed triangles in $\tau_O(\triangle_a) \cup \tau_R(\triangle_a)$, except $\triangle$, from left to right until we find an apexed triangle $\triangle'$ such that there is a point on $\triangle_a \cap \triangle' \cap uv$ which is equidistant from $\mathrm{D}(\triangle_a)$ and s_k. Then we apply the procedure in (1) with $\triangle'$ instead of $\triangle$. If there is no such apexed triangle, we have $U(s_k) = U'$ and $\tau_U(s_k) = \tau'_U$.

(3) Otherwise, we have $d(x, s_k) \geq d(x, \mathrm{D}(\triangle_a))$ for all points $x \in \triangle_a \cap \triangle \cap uv$. Then the distance function of $\triangle_a$ does not appear on $U(s_k)$. Thus, we remove $\triangle_a$ and its distance function from τ'_U and U', respectively. We consider the apexed triangles in $\tau_L(\triangle_a)$ from right to left. For an apexed triangle $\triangle' \in \tau_L(\triangle_a)$, we do the following. Since τ'_U is updated, we update $\triangle_a$ to the last element of τ'_U. Afterwards, we check whether $d(x, s_k) \geq d(x, \mathrm{D}(\triangle_a))$ for all points $x \in \triangle_a \cap \triangle' \cap uv$ if $\triangle'$ overlaps $\triangle_a$. If so, we remove $\triangle_a$ from τ'_U and update $\triangle_a$. We do this until we find an apexed triangle $\triangle' \in \tau_L(\triangle_a)$ such that this test fails. Then, there is a point on $\triangle' \cap \triangle_a \cap uv$ which is equidistant from $\mathrm{D}(\triangle_a)$ and s_k. After we reach such an apexed triangle $\triangle'$, we apply the procedure in (1) with $\triangle'$ instead of $\triangle$.

Case 2 : No apexed triangle in τ_k overlaps with $\triangle_a$. If $\tau_O(\triangle_a) = \phi$, we cannot compare the distance function of any apexed triangle in τ_k with the distance function of $\triangle_a$ directly, so we need a different method to handle this.

There are two possible subcases: either $\tau_L(\triangle_a) = \phi$ or $\tau_R(\triangle_a) = \phi$. Note that these are the only possible subcases since the union of the apexed triangles with the same definer is connected. For the former subcase, the upper envelope of sites from s_1 to s_k is discontinuous at the right endpoint of the bottom side of $\triangle_a$. Thus $g_\triangle$ does not appear on $U(s_k)$ for any apexed triangle $\triangle \in \tau_k$. Thus $U(s_k) = U'$ and $\tau_U(s_k) = \tau'_U$.

For the latter subcase, at most one of s_k and $\mathrm{D}(\triangle_a)$ has its Voronoi cell in $\mathsf{FVD}[S_k]$, where $S_k = \{s_1, \ldots, s_k\}$, by Lemma 2. In this case, we can find a site (s_k or $\mathrm{D}(\triangle_a)$) which

does not have its Voronoi cell in $\mathsf{FVD}[S_k]$ in constant time once we maintain $\pi(s, x_1), \pi(s, x_2)$ and their geodesic distances during the whole procedure (for all cases), where s is the site we are considering, which changes from s_1 to s_ℓ, and x_1, x_2 are the left and the right endpoints of the bottom side of the last apexed triangle $\triangle_a$ on τ'_U. We describe this procedure at the end of this subsection.

If s_k does not have its Voronoi cell in $\mathsf{FVD}[S_k]$, then $U(s_k) = U'$ and $\tau_U(s_k) = \tau'_U$. If $\mathrm{D}(\triangle_a)$ does not have its Voronoi cell in $\mathsf{FVD}[S_k]$, we remove all apexed triangles with definer $\mathrm{D}(\triangle_a)$ from τ'_U and their distance functions from U'. Since such apexed triangles lie at the end of τ'_U consecutively, it takes the time proportional to the number of the apexed triangles. Afterwards, we do this until the last element of τ_k and the last element of τ'_U overlap in their bottom sides. When the two elements overlap, we apply the procedure of Case 1.

In total, the running time is bounded by $O(|A|)$.

Subcase of Case 2 : $\tau_O(\triangle_a) = \phi$ and $\tau_R(\triangle_a) = \phi$. To handle this subcase, we maintain $\pi(s, x_1), \pi(s, x_2)$ and their geodesic distances during the whole procedure (for all cases), where s is the site we are considering, which changes from s_1 to s_ℓ, and x_1, x_2 are the left and the right endpoints of the bottom side of the last apexed triangle $\triangle_a$ on τ'_U. Note that the three points s, x_1, x_2 and the apexed triangle $\triangle_a$ change while the procedure is executed. Whenever they change, we update $\pi(s, x_1)$ and $\pi(s, x_2)$ using the previous geodesic paths. Then, we can compare $d(s_k, x)$ and $d(\mathrm{D}(\triangle_a), x)$ for $x \in \triangle_a \cap uv$ in constant time for this subcase, thus we can decide whether s_k or $\mathrm{D}(\triangle_a)$ does not have its Voronoi cell in $\mathsf{FVD}[S_k]$ in constant time.

We will show that updating the geodesic paths takes $O(|A|)$ time in total. We show this for $\pi(s, x_2)$ first. Let H_{uv} denote the region bounded by uv, $\pi(v, \mathrm{N}(u))$, $C[\mathrm{N}(v), \mathrm{N}(u)]$ and $\pi(\mathrm{N}(v), u)$, which is called an *hourglass* of uv in [2]. The total complexity of H_e for all transition edges e is $O(n)$ and they can be computed in $O(n)$ time as shown in [2]. Moreover, they show that the complexity of H_{uv} is bounded by $O(|A|)$. Thus the shortest path trees rooted at u and v in H_{uv} can be computed in $O(|A|)$ time and the total complexity of the shortest path trees is $O(|A|)$ [8]. This is done only once during the whole procedure.

Once the shortest path trees rooted at u and v are computed, the update time is bounded by the number of edges added to or removed from the previous geodesic path. Note that the edges in $\pi(s, x_2)$, except the edge adjacent to x_2, are also edges of the shortest path trees. In the following, we show that each edge of the shortest path trees is removed from $\pi(s_k, x)$ at most $O(1)$ times while we update $\pi(s, x_2)$, which implies that the update time is $O(|A|)$ time in total.

We already have $\pi(s_k, x_2)$ and we are to compute $\pi(s'_k, x'_2)$. There are three different cases; (1) $s'_k = s_k$ and x'_2 lies to the right of x_2, (2) $s'_k = s_k$ and x'_2 lies to the left of x_2 ($x'_2 = x_1$), and (3) $s'_k = s_{k+1}$ and $x'_2 = x_2$.

The first case occurs only when we add a new apexed triangle into τ'_U. Moreover, when we add one new apexed triangle into τ'_U, at most one edge of $\pi(s_k, x_2)$ is removed since the definer of the new apexed triangle is $s_k = s'_k$. Thus, the number of deletions of the edges in $\pi(s, x_2)$ corresponding to the first case is bounded by $O(|A|)$.

The second case occurs only when we remove the last apexed triangle $\triangle_a$ in τ'_U. It is possible that we have to remove more than one edges from $\pi(s_k, x_2)$ when one apexed triangle is removed. See Figure 1(a). Assume that we have to remove more than one edges when $\triangle_a$ is removed. Then x'_2 and x_2 are the left and the right endpoints of the bottom side of $\triangle_a$, respectively. Let w_1w_2 be an edge in $\pi(s_k, x_2)$ which is not adjacent to x_2 and is not in $\pi(s_k, x'_2)$. Let w'_1 and w'_2 be the two points on ∂P hit by the rays from w_1w_2 towards

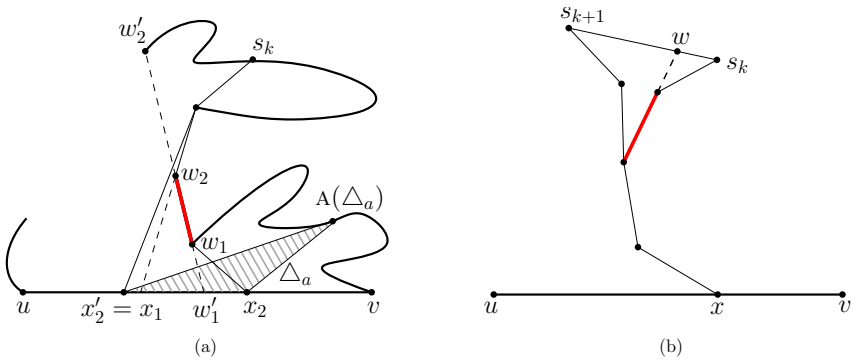

Figure 1 (a) When the gray apexed triangle is removed from τ'_U, we remove two edges (other than w_1x_2) from $\pi(s_k, x_2)$ to obtain $\pi(s_k, x'_2)$. After the gray apexed triangle is removed, no apexed triangle whose bottom side lies in x'_2v has its apex on $C[w_2, v]$. Thus the red thick line segment is not removed anymore. (b) The red thick line segment appears on $\pi(s, x_2)$ for some $x_2 \in uv$ only if $s \in C[w, v]$. Therefore, the red line does not appear on $\pi(s_j, x_2)$ for any $x_2 \in uv$ and any $j \geq k + 1$.

w_1 and w_2, respectively. Since $w'_1 \in x'_2x_2$, $\mathrm{A}(\triangle_a)$ lies in $C[w_1, v]$ by the construction of the apexed triangles and the fact that the definers of apexed triangles in τ'_U lie in $C[s_k, \mathrm{N}(u)]$. Note that once $\triangle_a$ is removed from τ'_U, no apexed triangle with apex in $C[w_1, v]$ is added to τ'_U again. This implies that w_1w_2 does not appear on $\pi(s, x_2)$ again in the remaining procedure for computing the upper envelope on uv. Thus, the number of deletions of the edges in $\pi(s, x_2)$ corresponding to the second case is also $O(|A|)$.

For the last case, $s'_k = s_{k+1}$ lies after s_k from s_1. It occurs when we finish the procedure for handling s_k. Once we consider the site s'_k, we do not consider any site from s_1 to s_k again. Consider an edge e removed from $\pi(s_k, x_2)$ due to the last case. Let w be the point on s_ks_{k+1} crossed by the extension of e. See Figure 1(b). If $\pi(s, x_2)$ contains e for some $s \in C[\mathrm{N}(v), \mathrm{N}(u)]$ and some $x_2 \in uv$, we have $s \in C[w, v]$. This means that once e is removed due to the last case, e does not appear on the geodesic path $\pi(s, x_2)$ again in the remaining procedure. Thus, the number of deletions of the edges in $\pi(s, x_2)$ corresponding to the last case is also $O(|A|)$.

The geodesic distance $\pi(s, x_1)$ can be computed analogously. For the third case, we can show this for $\pi(s, x_1)$ as a similar way for $\pi(s, x_2)$. The number of deletions of the edges in $\pi(s, x_1)$ corresponding to the first and the second cases is bounded by the number of deletions of the edges in $\pi(s, x_2)$, which is $O(|A|)$.

▶ **Theorem 6.** *The farthest-point geodesic Voronoi diagram of the vertices of a simple n-gon P restricted to the boundary of P can be computed in $O(n)$ time.*

4 Decomposing the polygon into smaller cells

Until now, we have computed $\mathsf{rFVD} \cap \partial P$ of size $O(n)$. We add the points in $\mathsf{rFVD} \cap \partial P$ to the vertex set of P, and apply the algorithm to compute the apexed triangles with respect to the vertex set of P again [2]. Note that now there is no transition edge. Thus all apexed triangles are disjoint in their bottom sides. We have the set of the apexed triangles sorted along ∂P with respect to their bottom sides.

A subset A of P is *geodesically convex* if $\pi(x, y) \subseteq A$ for any $x, y \in A$. A vertex v of a simple polygon is *convex* (or *reflex*) if the angle at v is at most (or at least) π. We define a

t-path-cell for some $t \in \mathbb{N}$ as a simple polygon contained in P with all vertices on ∂P which is geodesically convex and contains at most t convex vertices.

In the following, for a cell C, $|\partial C|$ denotes the number of edges of C. For a curve γ, $|\mathsf{rFVD} \cap \gamma|$ denotes the number of the refined cells intersecting γ.

Sketch of the algorithm We subdivide P into t-path-cells recursively for some $t \in \mathbb{N}$ until each cell becomes a base cell. There are three types of base cells. The first type is a quadrilateral crossed by exactly one arc of rFVD through two opposite sides, which we call an *arc-quadrilateral*. The second type is a 3-path-cell. Note that a 3-path-cell is a pseudo-triangle. The third type is a region of P whose boundary consists of one convex chain and one concave chain, which we call a *lune-cell*. Note that a convex polygon is a lune-cell whose concave chain is just a vertex of the polygon.

Let $\{t_k\}$ be the sequence such that $t_1 = n$ and $t_k = \lfloor \sqrt{t_{k-1}} \rfloor + 1$. Initially, P itself is a t_1-path-cell. Assume that the kth iteration is completed. We show how to subdivide each t_k-path-cell with $t_k > 3$ into t_{k+1}-path-cells and base cells in the $(k+1)$th iteration in Section 4.1. Note that a base cell is not subdivided further.

While subdividing the polygon into cells, we compute the refined farthest-point geodesic Voronoi diagram restricted to the boundary of each cell C (of any kind) in time linear on $|\partial C|$ and $|\mathsf{rFVD} \cap \partial C|$. In Section 5, we show how to compute the refined farthest-point geodesic Voronoi diagram restricted to a base cell T in $O(|\mathsf{rFVD} \cap \partial T|)$ time once $\mathsf{rFVD} \cap \partial T$ is computed.

Properties and analysis Note that $t_k \leq 3$ with $k = c \log\log n$ for some constant $1 < c \in \mathbb{R}$.[1] Moreover, in the kth iteration, P is subdivided into t_k-path-cells and base cells. Thus, in $O(\log\log n)$ iterations, every t-path-cell gets subdivided into base cells.

We will show that each iteration takes $O(n)$ time in Section 4.1, which implies that the overall running time for the computation in this section is $O(n \log\log n)$. After the last iteration of the subdivision, there are $O(n \log\log n)$ base cells. See Corollary 14. Moreover, an arc of rFVD intersects $O(1)$ t-path-cells, arc-quadrilaterals, and lune-cells created in the same iteration. See the proof of Lemma 12. With these facts, we will show that the total complexity of the refined farthest-point geodesic Voronoi diagram restricted to the boundaries of all cells in the kth iteration is $O(kn)$ for any $k \in \mathbb{N}$. See Lemma 13.

4.1 Subdividing a t-path-cell into smaller cells

In this section, we subdivide each t_k-path-cell into t_{k+1}-path-cells and base cells. If a t_k-path-cell C is a pseudo-triangle or a lune-cell, C is a base cell and we do not subdivide it further. Otherwise, we subdivide it using the algorithm in this section.

The subdivision consists of three phases. In the first phase, we subdivide each t_k-path-cell into t_{k+1}-path-cells by a curve connecting at most t_{k+1} vertices of the t_k-path-cell. In the second phase, we subdivide each t_{k+1}-path-cell further along an arc of rFVD crossing the cell if there is such an arc. In the last phase, we subdivide cells created in the second phase into t_{k+1}-path-cells and lune-cells.

[1] $t_k \leq 2\sqrt{t_{k-1}} \leq 2\sqrt{2\sqrt{t_{k-2}}} \leq \ldots \leq 2^{1+1/2+1/4+\ldots+(1/2)^k} n^{(1/2)^k} \leq 2^2 n^{(1/2)^{c \log\log n}} \leq O(n^{(\log n)^{1/c}}) \leq O(n^{\log n}) \leq O(1)$ for $k = c \log\log n$.

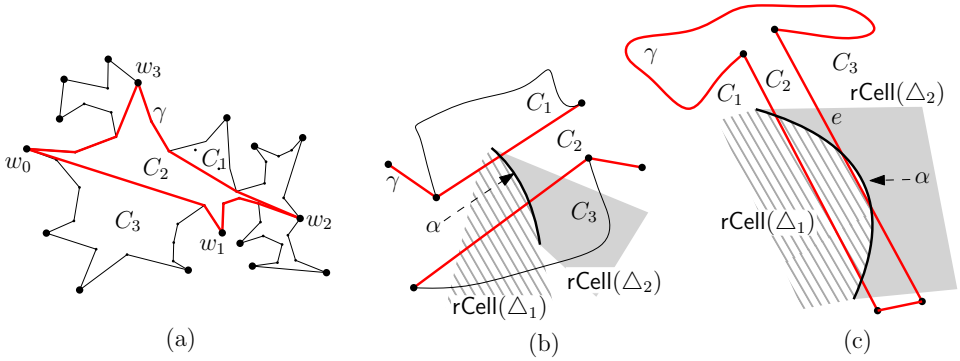

Figure 2 (a) The region bounded by the black curve is a 16-path-cell. All convex vertices are marked with black disks. The region is subdivided into six 5-path-cells by the red thick curve consisting of $\pi(w_0, w_1), \pi(w_1, w_2), \pi(w_2, w_3)$ and $\pi(w_3, w_0)$. (b) The arc α of rFVD intersects C_1, C_2, C_3 and crosses C_2. (c) The arc α of rFVD intersects C_1, C_2, C_3 and crosses C_2. Note that α does not cross C_3.

4.1.1 Phase 1. Subdivision by a curve connecting at most t_{k+1} vertices

Let C be a t_k-path-cell computed in the kth iteration. Recall that C consists of at most t_k convex vertices on its boundary and is simple. Let β be the largest integer satisfying that $\beta\lfloor\sqrt{t_k}\rfloor$ is less than the number of the convex vertices of C. Then we have $\beta \leq \lfloor\sqrt{t_k}\rfloor + 1 = t_{k+1}$.

We choose $\beta + 1$ vertices $w_0, w_1, \ldots, w_\beta$ from the convex vertices of C as follows. We choose an arbitrary convex vertex of C and denote it by w_0. Then we choose the $j\lfloor\sqrt{t_k}\rfloor$th convex vertex of C from w_0 in clockwise order and denote it by w_j for all $j = 1, \ldots, \beta$. We set $w_{\beta+1} = w_0$. Then we construct the closed curve γ_C (or simply γ when C is clear from context) consisting of the geodesic paths $\pi(w_0, w_1), \pi(w_1, w_2), \ldots, (w_\beta, w_0)$. See Figure 2(a). In other words, the closed curve γ_C is the boundary of the geodesic convex hull of $w_0, \ldots, w_\beta$. Note that γ is *simple*, that is, it does not cross itself. Moreover, γ is contained in C since C is geodesically convex.

We compute γ in time linear to the number of edges of C as follows. The algorithm computing geodesic paths in [11] takes k source-destination pairs as input, where both sources and destinations are on the boundary of a simple polygon. It returns the geodesic path between the source and the destination for each input pair. For all pairs, computing the geodesic paths takes $O(N + k)$ time in total if k shortest paths do not cross (but possibly overlap) one another, where N is the complexity of the polygon. In our case, the pairs (w_j, w_{j+1}) for $j = 0, \ldots, \beta$ are $\beta + 1$ input source-destination pairs. Since the geodesic paths for all input pairs do not cross one another, γ can be computed in $O(\beta + |\partial C|) = O(|\partial C|)$ time. Then we compute $\mathsf{rFVD} \cap \gamma$ in $O(|\mathsf{rFVD} \cap \partial C| + |\partial C|)$ time using $\mathsf{rFVD} \cap \partial C$ which has already been computed in the kth iteration. We will describe this procedure in Section 4.2.

The curve γ subdivides C into t_{k+1}-path-cells. The set $C \setminus \gamma$ consists of at least $\beta + 2$ connected components. Note that the closure of each connected component is a t_{k+1}-path-cell. Moreover, the union of the closures of all connected components is exactly C since C is simple. These components define the *subdivision of C induced by γ*.

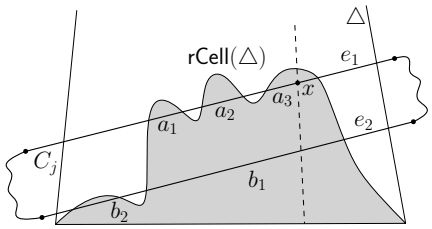

Figure 3 $\mathsf{rCell}(\triangle) \cap \partial C_j$ consists of five connected components (line segments) a_i contained in e_1 and b_j contained in e_2 for $i = 1, 2, 3$ and $j = 1, 2$.

4.1.2 Phase 2. Subdivision along an arc of rFVD

After subdividing C into t_{k+1}-path-cells $C_1, \ldots, C_\delta$ ($\delta \geq \beta + 2$) by the curve γ_C, an arc α of rFVD may cross C_j for some $1 \leq j \leq \delta$. In the second phase, for each arc α crossing C_j, we isolate the subarc $\alpha \cap C_j$ from C_j by creating a new cell which we call an arc-quadrilateral. For an arc-quadrilateral $\square$ created by an arc α, we have $\mathsf{rFVD} \cap \square = \alpha \cap C_j$.

▶ **Lemma 7.** *For a geodesically convex polygon C with t convex vertices ($t \in \mathbb{N}$), let $\tilde{\gamma}$ be a simple closed curve connecting $t' \leq t$ convex vertices of C lying on ∂P such that two consecutive vertices in clockwise order are connected by a geodesic path. Then, for each arc α of rFVD with $\alpha \cap C \neq \phi$, α intersects at most three cells in the subdivision of C by $\tilde{\gamma}$.*

Proof. Consider an arc α of rFVD with $\alpha \cap C \neq \phi$. The arc α is part of either the bisector of two sites or a side of some apexed triangle.

For the first case, α is part of a hyperbola. Let s_1 and s_2 be the two sites equidistant from any point in α. The combinatorial structure of the geodesic path from s_1 (or s_2) to any point in α is the same. This means that α is contained in the intersection of two apexed triangles $\triangle_1, \triangle_2$, one with definer s_1 and the other with definer s_2. Note that $\triangle_1 \cap \triangle_2$ intersects $\tilde{\gamma}$ at most twice and contains no vertex of $\tilde{\gamma}$ in its interior. Thus, $\triangle_1 \cap \triangle_2$ intersects at most two edges e_1, e_2 of $\tilde{\gamma}$, and so does α. For a cell C' in the subdivision of C by $\tilde{\gamma}$, α intersects C' if and only if C' contains e_1 or e_2 on its boundary.

Thus there exist at most three such cells in the subdivision of C by $\tilde{\gamma}$ and the lemma holds for the first case. See Figure 2(b)(c). For the second case, the arc α is a line segment. Thus α intersects at most three cells in the subdivision of C by $\tilde{\gamma}$. ◀

Since C is geodesically convex, α intersects at most two edges of a cell C in the first phase, which can be proved in a way similar to the proof of Lemma 7. We say an arc α of rFVD *crosses* a cell C' if exactly two edges of C' intersect α. For example, in Figure 2(c), α crosses C_2 while α does not cross C_3 because there is only one edge of C_3 intersecting α.

Consider $\mathsf{rCell}(\triangle) \cap \partial C_j$ for an apexed triangle $\triangle$. It is possible that $\mathsf{rCell}(\triangle) \cap \partial C_j$ consists of more than one connected components. See Figure 3. Each connected component of $\mathsf{rCell}(\triangle) \cap \partial C_j$ is a line segment.

First, we find an arc α of rFVD that crosses C_j. Since the points in $\mathsf{rFVD} \cap \partial C_j$ along ∂C_j have already been computed, we can scan them in clockwise order. For all arcs, it can be done in $O(|\mathsf{rFVD} \cap C_j|)$ time by the following two lemmas.

▶ **Lemma 8.** *There is an arc α of rFVD crossing C_j if and only if there are two distinct apexed triangles $\triangle_1$ and $\triangle_2$, and two connected components (line segments) of $\mathsf{rCell}(\triangle_1) \cap \partial C_j$ such that each of the connected components is adjacent to a connected component (line segment) of $\mathsf{rCell}(\triangle_2) \cap \partial C_j$.*

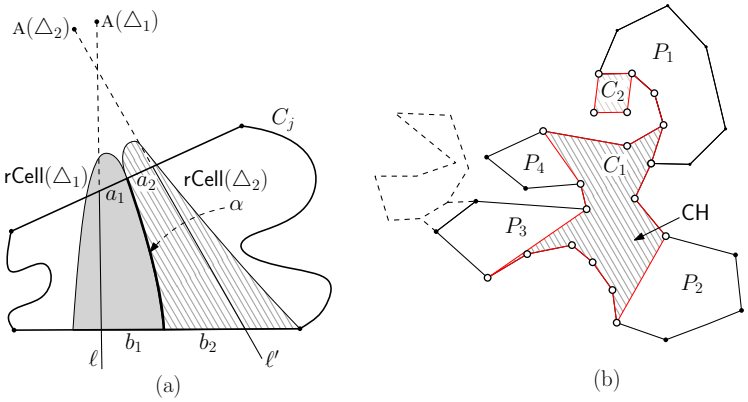

Figure 4 (a) The arc α of rFVD crosses C_j, thus we isolate this by creating the arc-quadrilateral bounded by ℓ_1, ℓ_2 and ∂C_j. (b) The vertices marked with empty disks lie on ∂P while the other vertices lie in $\text{int}(P)$. We subdivide the cell into one t-path-cell CH and four lune-cells $P_1, \ldots, P_4$.

Proof. Let α be an arc of rFVD. Then there are two apexed triangles $\triangle_1$ and $\triangle_2$ such that $\mathsf{rCell}(\triangle_1)$ and $\mathsf{rCell}(\triangle_2)$ share their boundary on α. If α crosses C_j, $\alpha \cap \partial C_j$ contains at least two points where a connected component $\mathsf{rCell}(\triangle_1) \cap \partial C_j$ meets a connected component of $\mathsf{rCell}(\triangle_2) \cap \partial C_j$. See Figure 4(a).

Let $(\triangle_1, \triangle_2)$ be a pair of apexed triangles satisfying the condition in the lemma and x, y be two points where a connected component $\mathsf{rCell}(\triangle_1) \cap \partial C_j$ meets a connected component of $\mathsf{rCell}(\triangle_2) \cap \partial C_j$. Since both $\mathsf{rCell}(\triangle_1)$ and $\mathsf{rCell}(\triangle_2)$ are connected, there is a common boundary of them between x and y. Note that for any point p on the common boundary, we have $g_{\triangle_1}(p) = g_{\triangle_2}(p) > 0$ by definition. The common boundary is a hyperbolic arc which crosses C_j. ◀

▶ **Lemma 9.** *All arcs α of* rFVD *crossing C_j can be found in $O(|\mathsf{rFVD} \cap \partial C_j|)$ time. Moreover, the pair $(\triangle_1, \triangle_2)$ of apexed triangles corresponding to each α such that $\alpha \cap C_j = \{x \in C_j : g_{\triangle_1}(x) = g_{\triangle_2}(x) > 0\}$ can be found in the same time.*

Proof. For each apexed triangle $\triangle$, we find all connected components of $\mathsf{rCell}(\triangle) \cap \partial C_j$. It can be done in $O(|\mathsf{rFVD} \cap C_j|)$ time for all apexed triangles. Since $\text{int}(\triangle)$ intersects ∂C_j at most twice and contains no vertex of C_j, at most two edges e_1, e_2 of ∂C_j contain all connected components of $\mathsf{rCell}(\triangle) \cap \partial C_j$. See Figure 3. Without loss of generality, we assume that e_1 contains the point $x \in \mathsf{rCell}(\triangle) \cap \partial C_j$ closest to $\text{A}(\triangle)$. We insert all connected components of $\mathsf{rCell}(\triangle) \cap e_1$ in the clockwise order along ∂C_j into a queue. Afterwards, we consider the connected components of $\mathsf{rCell}(\triangle) \cap e_2$ in the clockwise order along ∂C_j one by one.

To handle a connected component r of $\mathsf{rCell}(\triangle) \cap e_2$, we do the following. Let x be a point in the first element r' of the queue. If the line passing through x and $\text{A}(\triangle)$ intersects r, then we remove r' from the queue and check whether r and r' are incident to the same refined cell $\mathsf{rCell}(\triangle')$. If so, we compute an arc corresponding to $\triangle$ and $\triangle'$ and return the arc and the pair $(\triangle, \triangle')$. We repeat this until the line passing through a point of the first element of the queue does not intersect r. Then, we handle the connected component of $\mathsf{rCell}(\triangle) \cap e_2$ next to r.

For each arc α crossing C_j, there are the two connected components r, r' of $\mathsf{rFVD}(\triangle) \cap C_j$ incident to α. Without loss of generality, we assume that r contains the point in $r \cup r'$ closest to $\text{A}(\triangle)$. Since $\alpha \cup r$ is a connected curve, the set of the points where the line passing

through some point $x \in \alpha \cup r$ and $\mathrm{A}(\triangle)$ intersects ∂C_j is connected. Moreover, the set is contained in $\mathsf{rCell}(\triangle)$ by Lemma 4. Since r' is contained in the set, the line passing through a point in r and $\mathrm{A}(\triangle)$ intersects r'. Thus, the procedure finds α. ◀

Note that $\alpha \cap C_j$ consists of at most two connected components. For the case that it consists exactly two connected components, we consider each connected component separately. Thus we show the case that $\alpha \cap C_j$ is connected.

For an arc α crossing C_j, we subdivide C_j further into two cells with t' convex vertices for $t' \leq t_{k+1}$ and one arc-quadrilateral by adding two line segments bounding α such that no arc other than α intersects the arc-quadrilateral. Let $(\triangle_1, \triangle_2)$ be the pair of apexed triangles corresponding to α. Let a_1, b_1 (and a_2, b_2) be the two connected components of $\mathsf{rCell}(\triangle_1) \cap \partial C_j$ (and $\mathsf{rCell}(\triangle_2) \cap \partial C_j$) incident to α such that a_1, a_2 are adjacent to each other and b_1, b_2 are adjacent to each other. See Figure 4(a).

Without loss of generality, we assume that a_1 is closer than b_1 to $\mathrm{A}(\triangle_1)$. Let x be any point on a_1. Then the farthest neighbor of x is the definer of $\triangle_1$. We consider the line ℓ_1 passing through x and the apex of $\triangle_1$. Then the intersection between C_j and ℓ_1 is contained in the closure of $\mathsf{rCell}(\triangle_1)$ by Lemma 4. Similarly, we find the line ℓ_2 passing through the apex of $\triangle_2$ and a point on a_2.

We subdivide C_j into two cells with at most t_{k+1} convex vertices and one arc-quadrilateral by two lines ℓ_1 and ℓ_2. The quadrilateral bounded by the two lines and ∂C_j is an arc-quadrilateral since α crosses the quadrilateral but no other arcs of rFVD intersect the quadrilateral.

We do this for all arcs crossing some C_j. Note that no arc crosses the resulting cells other than arc-quadrilaterals by the construction. Then the resulting cells with at most t_{k+1} convex vertices and arc-quadrilaterals are the cells in the subdivision of C in the second phase.

▶ **Lemma 10.** *No arc of* rFVD *crosses cells other than arc-quadrilaterals created in the second phase of the subdivision.*

4.1.3 Phase 3. Subdivision by a geodesic convex hull

Note that some cell C' with t' convex vertices for $3 < t' \leq t_{k+1}$ created in the second phase might be neither a t'-path-cell nor a base cell since some vertices of the cell might be in $\mathrm{int}(P)$. In the third phase, we subdivide such cells further into t'-path-cells and base cells.

To subdivide C' into t_{k+1}-path-cells and base cells, we first compute the geodesic convex hull CH of the vertices of C' lying on ∂P in time linear to the number of edges in C' using the algorithm for computing k-pair shortest paths in [11]. Consider the connected components of $C' \setminus \partial\mathsf{CH}$. There are two types of the connected components. A connected component of the first type is enclosed by a closed simple curve which is part of $\partial\mathsf{CH}$. For example, C_1 and C_2 in Figure 4(b) are the connected components belonging to this type. A connected component of the second type is enclosed by a subchain of $\partial\mathsf{CH}$ from u to w in clockwise order and a subchain of $\partial C'$ from w to u in counterclockwise order for some $u, w \in \partial P$. For example, P_i in Figure 4(b) is the connected component belonging to the second type for $i = 1, \ldots, 4$.

By the construction, a connected component belonging to the first type has all its vertices on ∂P. Moreover, it has at most t' convex vertices since C' has t' convex vertices. Therefore, the closure of a connected component of $C' \setminus \partial\mathsf{CH}$ belonging to the first type is a t'-path-cell.

Every vertex of C' lying in $\mathrm{int}(P)$ is convex with respect to C' by the construction of C'. Thus, for a connected component P' belonging to the second type, the part of $\partial P'$ from $\partial C'$ is a convex chain with respect to P'. Moreover, the part of $\partial P'$ from $\partial\mathsf{CH}$ is the geodesic

path between two points, thus it is a concave chain with respect to P'. Therefore, the closure of a connected component belonging to the second type is a lune-cell.

Since C' is a simple polygon, the union of the closures of all connected components of $C' \setminus \mathsf{CH}$ is exactly C'. The closures of all connected components belonging to the first and the second types are t_{k+1}-path-cells and lune-cells created in the last phase of the $(k+1)$th iteration, respectively. We compute the t_{k+1}-path-cells and the lune-cells subdivided by $\partial\mathsf{CH}$. Then, we compute $\mathsf{rFVD} \cap \partial\mathsf{CH}$ using the procedure in Section 4.2.

The resulting t_{k+1}-path-cells and base cells are the final decomposition of C of the $(k+1)$th iteration.

4.1.4 Analysis of the complexity

We first bound the complexity of the refined farthest-point geodesic Voronoi diagram restricted to the boundary of the cells in each iteration. The two following technical lemmas are used to bound the complexity.

▶ **Lemma 11.** *Let $\triangle_1, \triangle_2$ be two apexed triangles such that $\mathsf{rCell}(\triangle_1)$ and $\mathsf{rCell}(\triangle_2)$ share an arc α of rFVD on their boundaries. Then any two line segments tangent to α whose endpoints lie outside of $\triangle_1 \cup \triangle_2$ intersect at some point other than their endpoints.*

Proof. If α is a line segment, then it is part of the common boundary of $\triangle_1$ and $\triangle_2$. Thus, the lemma holds. Now, we consider the case that α is hyperbolic.

Without loss of generality, we assume that the line containing $\mathrm{A}(\triangle_1)$ and $\mathrm{A}(\triangle_2)$ is the x-axis. Note that α is part of a hyperbola whose foci lie on the x-axis. Moreover, α does not intersect the x-axis. Let h_1 and h_2 be the lines tangent to α at the endpoints of α. We denote the region bounded by h_1, h_2 and α by R. Then it suffices to show that R is contained in $\triangle_1 \cap \triangle_2$.

Assume to the contrary that R is not contained in $\triangle_1 \cap \triangle_2$. If R is not contained in $\triangle_1$, then one of the sides of $\triangle_1$ incident to $\mathrm{A}(\triangle_1)$ intersects R. Thus, there is a line passing through $\mathrm{A}(\triangle_1)$ which intersects α twice, which is a contradiction by Lemma 4. The case that R is not contained in $\triangle_2$ is analogous. ◀

▶ **Lemma 12.** *An arc α of rFVD intersects at most nine t_k-path-cells and $O(k)$ base cells at the end of the kth iteration. Moreover, there are at most three t_k-path-cells that α intersects but does not cross at the end of the kth iteration.*

Proof. Let α be an arc of rFVD. By Lemma 11, there is at most one t_k-path-cell at the end of the kth iteration that α intersects but does not cross and no endpoint of α is contained in. Therefore, there are at most three t_k-path-cells that α intersects but does not cross at the end of the kth iteration.

Now, we show that α intersects at most nine t_k-path-cells and $O(k)$ base cells at the end of the kth iteration. For $k = 1$, P itself is the decomposition of P, thus there exists only one cell.

For $k \geq 2$, assume that the lemma holds for the k'th iterations for all $k' \leq k$. Thus α intersects at most nine t_k-path-cells at the end of the kth iteration by the assumption. Therefore, at the end of the first phase of the $(k+1)$th iteration, α crosses at most 27 t_{k+1}-path-cells by Lemma 7. Note that $\alpha \cap C'$ might consist of two connected components for a t_{k+1}-path-cell C' created in the first phase. See Figure 2(c). In this case, we create two arc-quadrilaterals. If $\alpha \cap C'$ is connected, we create one arc-quadrilateral. Thus, we create at most 54 arc-quadrilaterals crossed by α in the second phase.

Recall that there are at most three t_k-path-cells that α intersects but does not cross at the end of the kth iteration. Let C_1, C_2, and C_3 be such t_k-path-cells. Note that if α crosses a t_k-path-cell $\hat{C}$, α crosses all t_{k+1}-path-cells in the subdivision of $\hat{C}$ of the first phase which α intersects. Then the algorithm constructs an arc-quadrilateral and isolates part of α from the remaining cells in the subdivision of $\hat{C}$ in the second phase. Thus it suffices to consider only C_1, C_2, C_3 to bound the number of t_{k+1}-path-cells and base cells that α intersects at the end of the $(k+1)$th iteration.

Recall that C_i is subdivided into t_{k+1}-path-cells and base cells by the geodesic convex hull CH of the vertices of C_i lying on ∂P for $i = 1, 2, 3$. By Lemma 7, α intersects at most three (any kinds of) cells in the subdivision of C_i in the last phase of the $(k+1)$th iteration. Thus, α intersects at most nine lune-cells and at most nine t_{k+1}-path-cells.

Thus, α intersects at most $O(1)$ base cells and nine t_{k+1}-path-cells at the end of the $(k+1)$th iteration, which implies the lemma. ◀

Now we are ready to bound the complexities of the cells and rFVD restricted to the cells in each iteration. Then we finally prove that the running time of the algorithm in this section is $O(n \log\log n)$.

▶ **Lemma 13.** *At the end of the kth iteration, the following holds.*
$\sum_{C:\text{a } t_k\text{-path-cell}} |\mathsf{rFVD} \cap \partial C| = O(n)$.
$\sum_{C:\text{a } t_k\text{-path-cell}} |\partial C| = O(n)$.
$\sum_{T:\text{a base cell}} |\mathsf{rFVD} \cap \partial T| = O(kn)$.
$\sum_{T:\text{a base cell}} |\partial T| = O(kn)$.

Proof. Let α be an arc of rFVD. The first and the third complexity bounds hold by Lemma 12 and the fact that the number of the arcs of rFVD is $O(n)$.

The second complexity bound holds since the set of all edges of the t_k-path-cells is a subset of the chords in some triangulation of P. Note that any triangulation of P has $O(n)$ chords. Moreover, each chord is incident to at most two t_k-path-cells.

For the last complexity bound, the number of edges of base cells whose endpoints are vertices of P is $O(n)$ since they are chords in some triangulation of P. Thus we count the edges of base cells which are not incident to a vertex of P. In the first phase, such an edge is not created. In the second phase, we create at most $O(1)$ such edges whenever we create one arc-quadrilateral. All edges created in the last phase have their endpoints on ∂P. Therefore, the total number of the edges of all base cells is asymptotically bounded by the number of arc-quadrilaterals, which is $O(kn)$. ◀

▶ **Corollary 14.** *In $O(\log\log n)$ iterations, the polygon is subdivided into $O(n \log\log n)$ base cells.*

▶ **Lemma 15.** *The subdivision in each iteration can be done in $O(n)$ time.*

Proof. In the first phase, we compute γ_C and $\mathsf{rFVD} \cap \gamma_C$ for each t-path-cell C from the previous iteration. The running time for this is linear to the total complexity of all t-path-cells in the previous iteration and rFVD restricted on the boundary of all t-path-cells, which is $O(n)$ by Lemma 13.

In the second phase, we first scan $\mathsf{rFVD} \cap C'$ for all cells C' from the first phase to find an arc crossing some cell. This can also be done in linear time by Lemma 9 and Lemma 13. For each arc crossing some t-path-cell, we compute two line segments bounding the arc and subdivide the cell into two smaller regions and one arc-quadrilateral in $O(1)$ time. Each arc crosses at most $O(1)$ cells from the first phase, and the time for this step is $O(n)$ in total.

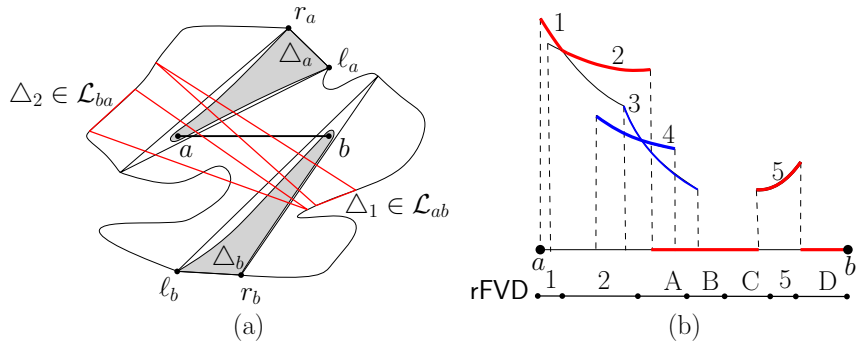

Figure 5 (a) The apexed triangles with their bottom sides in $C[\ell_a, r_b]$ and $C[\ell_b, r_a]$ are contained in $\mathcal{L}_{ab}$ and $\mathcal{L}_{ba}$, respectively. (b) The hyperbolic arcs are the distance functions associated with the apexed triangles in $\mathcal{L}_{ab}$. The set of red curves (1, 2, and 5) represent a partial upper envelope of all distance functions. Note that 3 and 4 do not appear on the complete upper envelope which coincides with rFVD. Thus, we do not need to consider them.

In the last phase, we further subdivide each cell which is not a base cell from the second phase. In the subdivision of a cell which is not a base cell C in the second phase, we first compute the geodesic convex hull CH of the vertices of C' lying on ∂P. The geodesic convex hull can be computed in time linear to the complexity of C'. By Lemma 18, $\mathsf{rFVD} \cap \partial\mathsf{CH}$ can be computed in $O(|\mathsf{rFVD} \cap \partial C'| + |\partial C'|)$ time. Note that all cells other than base cells from the second phase are interior disjoint. Moreover, the total number of the edges of such cells is $O(n)$. Similarly, the total complexity of $\mathsf{rFVD} \cap \partial C'$ for all such cells C' is $O(n)$. Therefore, the t-path-cells and lune-cells can be computed in $O(n)$ time. ◀

4.2 Computing rFVD restricted to the boundary of a t-path-cell

Remind that the bottom sides of all apexed triangles are interior-disjoint. Moreover, the union of them is ∂P by the construction. In this section, we describe a procedure to compute $\mathsf{rFVD} \cap \gamma_C$ in $O(|\mathsf{rFVD} \cap \partial C| + |\partial C|)$ time once $\mathsf{rFVD} \cap \partial C$ is computed. Recall that γ_C is the closed curve connecting consecutive points of every t_{k+1}th convex vertices of C in clockwise order.

If $\mathsf{rCell}(\triangle) \cap \gamma \neq \phi$ for an apexed triangle $\triangle$, then we have $\mathsf{rCell}(\triangle) \cap \partial C \neq \phi$. Thus, we consider only the apexed triangles $\triangle$ with $\mathsf{rCell}(\triangle) \cap \partial C \neq \phi$. Let $\mathcal{L}$ be the list of all such apexed triangles sorted along ∂P with respect to their bottom sides. Note that $|\mathcal{L}| = O(|\mathsf{rFVD} \cap \partial C|)$.

Consider a line segment ab contained in P. Without loss of generality, we assume that ab is horizontal and a lies to the left of b. Let $\triangle_a$ and $\triangle_b$ be the apexed triangles which maximize $g_{\triangle_a}(a)$ and $g_{\triangle_b}(b)$, respectively. If there is a tie by more than one apexed triangles, we choose an arbitrary one of them. With the two apexed triangles, we define two sorted lists $\mathcal{L}_{ab}$ and $\mathcal{L}_{ba}$. Let $\mathcal{L}_{ab}$ be the sorted list of the apexed triangles in $\mathcal{L}$ which intersect ab and whose bottom sides lie from the bottom side of $\triangle_a$ to the bottom side of $\triangle_b$ in clockwise order along ∂P. Similarly, let $\mathcal{L}_{ba}$ be the sorted list of the apexed triangles in $\mathcal{L}$ which intersect ab and whose bottom sides lie from the bottom side of $\triangle_b$ to the bottom side of $\triangle_a$ in clockwise order along ∂P. See Figure 5(a).

The following lemma together with Section 4.2.1 gives a procedure to compute $\mathsf{rFVD} \cap ab$.

▶ **Lemma 16.** *Let C be a geodesically convex polygon and a, b be two points with $ab \subset C$.*

Given the two sorted lists $\mathcal{L}_{ab}$ and $\mathcal{L}_{ba}$, $\mathsf{rFVD} \cap ab$ can be computed in $O(|\mathcal{L}_{ab}| + |\mathcal{L}_{ba}|)$ time.

Proof. Recall that the upper envelope of $g_\triangle$ for all apexed triangles $\triangle \in \mathcal{L}_{ab} \cup \mathcal{L}_{ba}$ (simply, the upper envelope for $\mathcal{L}_{ab} \cup \mathcal{L}_{ba}$) coincides with $\mathsf{rFVD} \cap ab$ in its projection on ab by definition. Thus we compute the upper envelope for $\mathcal{L}_{ab} \cup \mathcal{L}_{ba}$. To this end, we compute a "partial" upper envelope of $g_\triangle$ on ab for all apexed triangles $\triangle \in \mathcal{L}_{ab}$. After we do this for the apexed triangles in $\mathcal{L}_{ba}$, we merge the two "partial" upper envelopes on ab to obtain the complete upper envelope of $g_\triangle$ on ab for all apexed triangles $\triangle \in \mathcal{L}_{ab} \cup \mathcal{L}_{ba}$.

A *partial upper envelope* for $\mathcal{L}_{ab}$ is the upper envelope for $A \subset \mathcal{L}_{ab}$ satisfying that $\triangle \in \mathcal{L}_{ab}$ belongs to A if $\mathsf{rCell}(\triangle) \cap ab \neq \phi$. See Figure 5(b). Thus the upper envelope of two partial upper envelopes, one for $\mathcal{L}_{ab}$ and one for $\mathcal{L}_{ba}$, is the complete upper envelope for $\mathcal{L}_{ab} \cup \mathcal{L}_{ba}$. Note that a partial upper envelope for $\mathcal{L}_{ab}$ (and $\mathcal{L}_{ba}$) is not necessarily unique.

In the following, we show how to compute one of the partial upper envelopes for $\mathcal{L}_{ab}$. A partial upper envelope for $\mathcal{L}_{ba}$ can be computed analogously. The complete upper envelope can be constructed in $O(|\mathcal{L}_{ab}| + |\mathcal{L}_{ba}|)$ time by scanning the two partial upper envelopes.

For any two apexed triangles $\triangle_1, \triangle_2 \in \mathcal{L}_{ab}$ such that the bottom side of $\triangle_1$ comes before the bottom side of $\triangle_2$ from the bottom side of $\triangle_a$ in clockwise order along ∂P, $\mathsf{rCell}(\triangle_1) \cap ab$ comes before $\mathsf{rCell}(\triangle_2) \cap ab$ from a if they exist. Otherwise, there is a point contained in $\mathsf{rCell}(\triangle_1) \cap \mathsf{rCell}(\triangle_2)$ by Lemma 4, which contradicts that all refined cells are pairwise disjoint. With this property, a partial upper envelope for $\mathcal{L}_{ab}$ can be constructed in a way similar to the procedure for computing $\mathsf{rFVD} \cap \partial P$ in Section 3.1. The difficulty here is that we must avoid maintaining geodesic paths as it takes $O(n)$ time, which is too much for our purpose.

We consider the apexed triangles in $\mathcal{L}_{ab}$ from $\triangle_a$ to $\triangle_b$ one by one as follows. Let U be the current partial upper envelope of the distance functions of apexed triangles from $\triangle_a$ to $\triangle'$ of $\mathcal{L}_{ab}$ and τ be the list of the apexed triangles whose distance functions restricted to ab appear on U in the order in which they appear on U. We show how to update U to a partial upper envelope of the distance functions of apexed triangles from $\triangle_a$ to $\triangle$, where $\triangle$ is the apexed triangle next to $\triangle'$ in $\mathcal{L}_{ab}$. Let $\triangle_r$ be the last element in τ and μ be the line segment contained in ab such that $g_{\triangle_r}(x) = U(x) > 0$ for every point $x \in \mu$.

There are three possibilities: (1) $\triangle \cap \mu \neq \phi$. In this case, we compare the distance functions of $\triangle$ and $\triangle_r$ on $\triangle \cap \mu$. Depending on the result, we update U and τ as we did in Section 3.1. (2) $\triangle \cap \mu = \phi$ and $\triangle \cap ab$ lies to the right of μ. We append $\triangle$ to τ at the end and update U accordingly. (3) $\triangle \cap \mu = \phi$ and $\triangle \cap ab$ lies to the left of μ. We have to use a method different from the one in Section 3.1 to handle this case. Here, contrast to the case in Section 3.1, $\triangle_r$ intersects $\triangle$. Thus, we can check whether $\mathsf{rCell}(\triangle) \cap ab = \phi$ or $\mathsf{rCell}(\triangle_r) \cap ab = \phi$ easily as follows. Consider the set $R = \triangle \cap \triangle_r$. The distance functions associated with $\triangle$ and $\triangle_r$ have positive values on R, thus we can compare the geodesic distances from $\mathrm{D}(\triangle)$ and $\mathrm{D}(\triangle_r)$ to a point in R. Depending on the result, we can check in constant time whether $\mathsf{rCell}(\triangle)$ and $\mathsf{rCell}(\triangle_r)$ intersect the connected regions $\triangle \setminus R$ and $\triangle_r \setminus R$ containing $\mathrm{A}(\triangle)$ and $\mathrm{A}(\triangle_r)$, respectively. If $\mathsf{rCell}(\triangle)$ does not intersect the connected region $\triangle \setminus R$ containing $\mathrm{A}(\triangle)$, then $\mathsf{rCell}(\triangle_r)$ does not intersect ab. This also holds for $\mathsf{rCell}(\triangle_r)$. Depending on the result, we apply the procedure in Section 3.1.

Recall that we do not append an apexed triangle $\triangle$ to τ only if $\mathsf{rCell}(\triangle) \cap ab = \phi$. Similarly, we remove some apexed triangle $\triangle$ from τ only if $\mathsf{rCell}(\triangle) \cap ab = \phi$. Thus, by definition, U is a partial upper envelope of the distance functions for $\mathcal{L}_{ab}$.

As we mention above, we do this also for $\mathcal{L}_{ba}$. Then we compute the upper envelope of the two resulting partial upper envelopes, which is the complete upper envelope for $\mathcal{L}_{ab} \cup \mathcal{L}_{ba}$. This takes $O(|\mathcal{L}_{ab}| + |\mathcal{L}_{ba}|)$ time. ◀

▶ **Corollary 17.** *Let C be a geodesically convex polygon and E be a set of line segments which are contained in C with $|E| = O(1)$. Then $\mathsf{rFVD} \cap ab$ for all $ab \in E$ can be computed in $O(|\partial C| + |\mathsf{rFVD} \cap \partial C|)$ time.*

Recall that $\mathcal{L}$ is the list of all apexed triangles $\triangle$ with $\mathsf{rCell}(\triangle) \cap \partial C \neq \phi$ sorted along ∂P with respect to their bottom sides. Note that an apexed triangle intersects at most two edges of γ_C. Thus, once we compute $\mathcal{L}_{ab}$ and $\mathcal{L}_{ba}$ for all edges ab of γ_C, we can compute $\mathsf{rFVD} \cap \gamma_C$ in $O(|\mathcal{L}|)$ time.

4.2.1 Computing $\mathcal{L}_{ab}$ and $\mathcal{L}_{ba}$ for all edges ab of γ

In this section, we show how to compute $\mathcal{L}_{ab}$ and $\mathcal{L}_{ba}$ for all edges ab of γ in $O(|\mathcal{L}| + |\partial C|)$ time. Recall that all endpoints of the geodesic paths bounding the t-path-cell C lie in ∂P. Let ab be an edge of γ, where b is the clockwise neighbor of a. The edge ab is a chord of P and divides P into two subpolygons such that $\gamma \setminus ab$ is contained in one of the subpolygons. Let $P_1(ab)$ be the subpolygon containing $\gamma \setminus ab$ and $P_2(ab)$ be the other subpolygon. For an apexed triangle in $\mathcal{L}_{ab}$, its bottom side lies in $\partial P_2(ab)$ and its apex lies in $\partial P_1(ab)$. On the other hand, for an apexed triangle in $\mathcal{L}_{ba}$, its bottom side lies in $\partial P_1(ab)$ and its apex lies in $\partial P_2(ab)$. Moreover, if its apex lies in $P_j(ab)$, then so does its definer for $j = 1, 2$. By the construction, $P_2(ab)$ and $P_2(e')$ are disjoint in their interior for any edge $e' \in \gamma \setminus \{ab\}$.

Using this, we compute $\mathcal{L}_{ab}$ for all edges ab in γ as follows. Initially, $\mathcal{L}_{ab}$ for all edges ab are set to ϕ. We update the list by scanning the apexed triangles in $\mathcal{L}$ from the first to the end. When we handle an apexed triangle $\triangle \in \mathcal{L}$, we first find the edge ab of γ such that $P_2(ab)$ contains the bottom side of $\triangle$ and check whether $\triangle \cap ab = \phi$. If it is nonempty, we append $\triangle$ to $\mathcal{L}_{ab}$. Otherwise, we do nothing. Then we handle the apexed triangle next to $\triangle$. For $\mathcal{L}_{ba}$, we do analogously except that we find the edge ab of γ such that $P_2(ab)$ contains the *definer* of $\triangle$.

Note that any three apexed triangles $\triangle_1, \triangle_2, \triangle_3 \in \mathcal{L}$ appear on $\mathcal{L}$ in the order of their definers (and their bottom sides) appearing on ∂P. Thus to find the edge ab of γ such that $P_2(ab)$ contains the definer (or the bottom side) of $\triangle$, it is sufficient to check at most two edges; the edge e' such that $P_2(e')$ contains the bottom side of the apexed triangle previous to $\triangle$ in $\mathcal{L}$ and the clockwise neighbor of e'. Therefore, this procedure takes in $O(|\mathcal{L}|)$ time.

The following lemmas summarize this section.

▶ **Lemma 18.** *Let C be a t-path-cell and $\tilde{\gamma}$ be a simple closed curve connecting $t' \leq t$ convex vertices of C lying on ∂P such that two consecutive vertices in clockwise order are connected by a geodesic path. Once $\mathsf{rFVD} \cap \partial C$ is computed, $\mathsf{rFVD} \cap \tilde{\gamma}$ can be computed in $O(|\mathsf{rFVD} \cap \partial C| + |\partial C|)$ time.*

▶ **Lemma 19.** *Each iteration takes $O(n)$ time and the algorithm in this section terminates in $O(\log \log n)$ iterations. Thus the running time of the algorithm in this section is $O(n \log \log n)$.*

5 Computing rFVD in the interior of a base cell

In this section, we consider a base cell T which is a lune-cell or a pseudo-triangle. Assume that $\mathsf{rFVD} \cap \partial T$ has already been computed. We extend $\mathsf{rFVD} \cap \partial T$ into the interior of the cell T in $O(|\mathsf{rFVD} \cap \partial T|)$ time.

To make the description easier, we first make two assumptions: (1) for any apexed triangle $\triangle$, $\mathsf{rCell}(\triangle) \cap \partial T$ is connected and contains the bottom side of $\triangle$, and (2) T is a lune-cell. At the end of this section, we generalize the algorithm to handle every base cell.

5.1 Definition for a new distance function

Without loss of generality, we assume that the bottom side of T is horizontal. We bound a domain by introducing a box B containing T because the algorithm in [5] is valid only when the domain is bounded. To apply the algorithm for computing the abstract Voronoi diagram in [5], we need to define a new distance function $f_\triangle : B \to \mathbb{R}$ since $g_\triangle$ is not continuous. Imagine that we partition B into five regions, as depicted in Figure 6(a), with respect to an apexed triangle $\triangle$. We will define $f_\triangle$ as a function consisting of at most five algebraic functions each of whose domain corresponds to a partitioned region in B.

Consider five line segments ℓ_1, ℓ_2, ℓ_3, ℓ_4 and ℓ_5 such that their common endpoint is $\mathrm{A}(\triangle)$ and the other endpoints lie on ∂B. The line segments ℓ_1 and ℓ_2 contain the left and the right corners of $\triangle$, respectively. The line segments ℓ_3 and ℓ_5 are orthogonal to ℓ_2 and ℓ_1, respectively. The line segment ℓ_4 is contained in the line bisecting the angle of $\triangle$ at $\mathrm{A}(\triangle)$ but it does not intersect $\mathrm{int}(\triangle)$. See Figure 6(a).

Then B is partitioned by these five line segments into five regions. We denote the region bounded by ℓ_1 and ℓ_2 which contains $\triangle$ by $G_{\mathrm{in}}(\triangle)$. Note that $\mathrm{D}(\triangle) \notin G_{\mathrm{in}}(\triangle)$ if $\mathrm{D}(\triangle) \neq \mathrm{A}(\triangle)$. The remaining four regions are denoted by $G_{\mathrm{Lside}}(\triangle)$, $G_{\mathrm{Ltop}}(\triangle)$, $G_{\mathrm{Rtop}}(\triangle)$, and $G_{\mathrm{Rside}}(\triangle)$ in the clockwise order from $G_{\mathrm{in}}(\triangle)$ along ∂B.

For a point $x \in G_{\mathrm{Lside}}(\triangle) \cup G_{\mathrm{Ltop}}(\triangle)$, let $\hat{x}_\triangle$ denote the orthogonal projection of x on the line containing ℓ_1. Similarly, for a point $x \in G_{\mathrm{Rside}}(\triangle) \cup G_{\mathrm{Rtop}}(\triangle) \setminus \ell_4$, let $\hat{x}_\triangle$ denote the orthogonal projection of x on the line containing ℓ_2. For a point $x \in G_{\mathrm{in}}(\triangle)$, we set $\hat{x}_\triangle = x$.

We define a new distance function $f_\triangle : B \to \mathbb{R}$ for each apexed triangle $\triangle$ with $\mathsf{rCell}(\triangle) \cap \partial T \neq \phi$ as follows.

$$f_\triangle(x) = \begin{cases} d(\mathrm{A}(\triangle), \mathrm{D}(\triangle)) - \|\hat{x}_\triangle - \mathrm{A}(\triangle)\| & \text{if } x \in G_{\mathrm{Ltop}}(\triangle) \cup G_{\mathrm{Rtop}}(\triangle) \\ d(\mathrm{A}(\triangle), \mathrm{D}(\triangle)) + \|\hat{x}_\triangle - \mathrm{A}(\triangle)\| & \text{otherwise,} \end{cases}$$

where $\|x - y\|$ denote the Euclidean distance between x and y. Note that $f_\triangle$ is continuous on B. Each contour curve, that is a set of points with the same function value, consists of two line segments and at most one circular arc. See Figure 6(b).

Here, we assume that there is no pair $(\triangle_1, \triangle_2)$ of apexed triangles such that two sides, one from $\triangle_1$ and the other from $\triangle_2$, are parallel. If there exists such a pair, the contour curves for two apexed triangles may overlap. We will show how to avoid the assumption at the end of this section by slightly perturbing the distance function defined in this section.

By the definition of $f_\triangle$, the following lemma holds.

▶ **Lemma 20.** *The difference of $f_\triangle(x_1)$ and $f_\triangle(x_2)$ is less than or equal to $\|x_1 - x_2\|$ for any two points $x_1, x_2 \in B$, where $\|x - y\|$ is the Euclidean distance between x and y.*

5.2 An algorithm for computing $\mathsf{rFVD} \cap T$

To compute the farthest-point geodesic Voronoi diagram restricted to T, we apply the algorithm in [5] with this new distance function, which computes the abstract Voronoi diagram in a domain where each site has a unique cell touching the boundary of the domain. In the abstract Voronoi diagram, no explicit sites or distance functions are given. Instead, for any pair of sites s, s', the open domains $D(s, s')$ and $D(s', s)$ are given. Let A be the set of all apexed triangles with $\mathsf{rFVD} \cap \partial T$. In our problem, we regard the apexed triangles in A as the sites and B as the domain for the abstract Voronoi diagram. For two apexed triangles $\triangle_1$

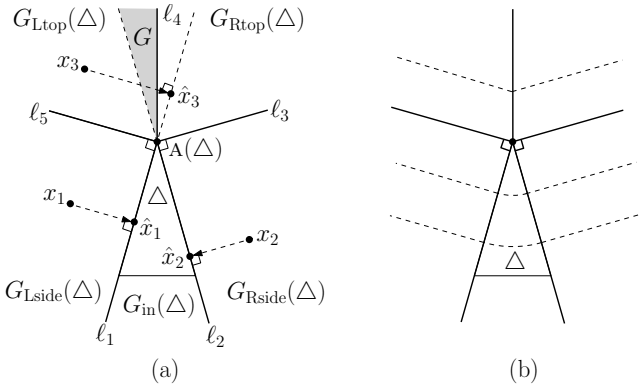

Figure 6 (a) Five regions are defined by the five line segments from ℓ_1 to ℓ_5 (b) The dashed curves are a few contour curves with respect to $f_{\triangle}$.

and $\triangle_2$ in A, we define the open domain $D(\triangle_1, \triangle_2)$ as the set $\{x \in B : f_{\triangle_1}(x) > f_{\triangle_2}(x)\}$. We denote the abstract Voronoi diagram for the apexed triangles by aFVD and the cell of $\triangle$ on aFVD by $\mathsf{aCell}(\triangle)$.

To apply the algorithm in [5], we show that the distance function we define in Section 5.1 satisfies the following. Let A' be a subset of A.

1. For any two apexed triangles $\triangle_1$ and $\triangle_2$ in A, the set $\{x \in B : f_{\triangle_1}(x) = f_{\triangle_2}(x)\}$ is a curve with endpoints on ∂B. The curve consists of $O(1)$ pieces of algebraic curves. (Lemma 28)
2. Each apexed triangle $\triangle$ in A' has exactly one connected and nonempty cell in the abstract Voronoi diagram of A'. (Corollary 26)
3. Each point in B belongs to the closure of an abstract Voronoi cell, which directly follows from the definition and Lemma 28.
4. The abstract Voronoi diagrams of A and A' form a tree and a forest, respectively. (Lemma 29)

Thus, we can compute aFVD using the algorithm in [5]. We will prove that aFVD restricted to T is exactly the refined farthest-point geodesic Voronoi diagram restricted to T. (See Corollary 24.) Note that we already have the abstract Voronoi diagram restricted to ∂T which coincides with the refined farthest-point geodesic Voronoi diagram restricted to ∂T. After computing aFVD on B, we traverse aFVD and extract aFVD lying inside T.

5.3 The abstract Voronoi diagram with the distance function

We assume that the first line segment of the geodesic path from $\text{A}(\triangle)$ to $\text{D}(\triangle)$ lies in the left side of the line containing ℓ_4. Remind that T is a lune cell and therefore it is bounded by a convex chain and a concave chain.

In this section, we show that the function we defined satisfies the conditions in Section 5.2. The following two technical lemmas are used to prove that $\mathsf{rCell}(\triangle)$ is contained in $\mathsf{aCell}(\triangle)$ for an apexed triangle $\triangle \in A$.

▶ **Lemma 21.** *Let $\triangle$ be an apexed triangle with $\mathsf{rCell}(\triangle) \cap T \neq \phi$ such that $\pi(\text{D}(\triangle), \text{A}(\triangle))$ does not overlap with (but possibly crosses) the concave chain of T. Then, $f_{\triangle}(x) \leq d(\text{D}(\triangle), x)$ for any point $x \in T$. The equality holds if x lies in $\triangle$.*

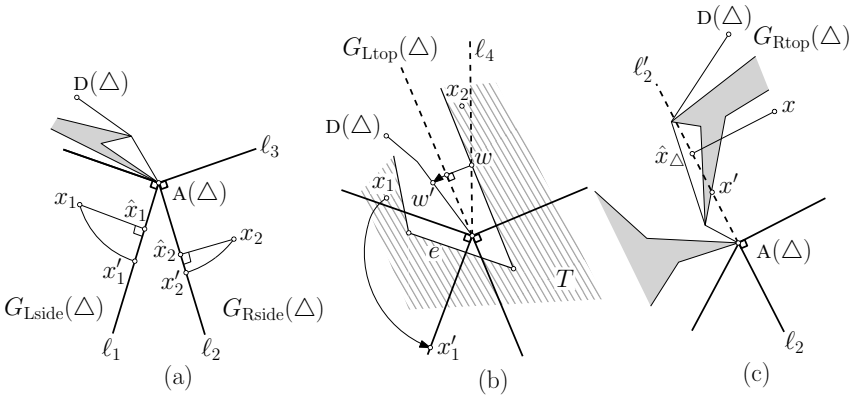

Figure 7 (a) $d(\text{D}(\triangle), \hat{x}_1) \leq d(\text{D}(\triangle), x_1')$ and $d(\text{D}(\triangle), \hat{x}_2) \leq d(\text{D}(\triangle), x_2')$. (b) $\pi(\text{D}(\triangle), x_1)$ contains $\text{A}(\triangle)$, thus $f_\triangle(x_1) < d(x_1, \text{D}(\triangle))$. For x_2, $\pi(\text{D}(\triangle), x_1)$ intersects e, thus either ∂T does not intersect ℓ_4 or intersects exactly once. (c) The point x' is $\triangle(x)$ or a point on ∂P. We prove the lemma for this case by showing that $f_\triangle(x) \leq f_\triangle(x') = d(\text{D}(\triangle), \text{A}(\triangle)) - \|x' - \text{A}(\triangle)\| < d(\text{D}(\triangle), x)$.

Proof. We first consider a point $x \in T \cap G_{\text{Lside}}(\triangle)$. Among all points on ℓ_1, $\text{A}(\triangle)$ is the point closest to $\text{D}(\triangle)$. Thus there exists a point x' on the line containing ℓ_1 with $d(\text{D}(\triangle), x) = d(\text{D}(\triangle), x')$. See Figure 7(a). Moreover, $d(\text{D}(\triangle), \hat{x}_\triangle) < d(\text{D}(\triangle), x')$, which implies that $f_\triangle(x) < d(\text{D}(\triangle), x)$ by definition. The case for a point in $G_{\text{Rside}}(\triangle)$ is analogous.

Consider a point $x \in T \cap G_{\text{Ltop}}(\triangle)$. Recall that x lies in T and ∂T consists of a convex chain and a concave chain. The set $T \setminus \triangle$ consists of two connected regions, one lies to the left of $\triangle$ and the other lies to the right of $\triangle$. There are two possibilities: (1) x lies in the left region or (2) x lies in the right region. See Figure 7(b). If x lies in the left region, $\pi(\text{D}(\triangle), x)$ contains $\text{A}(\triangle)$, thus $d(\text{D}(\triangle), x) > d(\text{D}(\triangle), \text{A}(\triangle)) > f_\triangle(x)$, which proves the lemma for this case. If x lies in the right region, $\pi(\text{D}(\triangle), x)$ contains an endpoint of e, where e is the edge of T crossing $\triangle$. Moreover, e is contained in the concave chain of T. Let w be the point where ℓ_4 intersects the concave chain of T. Note that $f_\triangle(x) < f_\triangle(w)$. Let w' be the point on $\pi(\text{D}(\triangle), \text{A}(\triangle))$ with $d(\text{A}(\triangle), w') = \|\text{A}(\triangle) - w\|$. If w' does not exist, then $f_\triangle(w) < 0$, thus $f_\triangle(x) < 0$, which implies $f_\triangle(x) < d(\text{A}(\triangle), x)$. Otherwise, $d(\text{D}(\triangle), w') < d(\text{D}(\triangle), a)$, where a is an endpoint of e. Note that $\pi(\text{D}(\triangle), x)$ contains at least one of the endpoints of e. This means that $f_\triangle(x) < f_\triangle(w) = d(\text{D}(\triangle), w') < d(\text{D}(\triangle), a) < d(\text{D}(\triangle), x)$, which proves the lemma for this case.

Consider a point $x \in T \cap G_{\text{Rtop}}(\triangle)$. Then $\hat{x}_\triangle$ is the orthogonal projection of x onto the line containing ℓ_2 by definition. Note that it is possible that the line segment ℓ_a connecting $\hat{x}_\triangle$ and $\text{A}(\triangle)$ intersects ∂P. To avoid this, we choose x' as the first point in $\ell_a \cap \partial P$ from $\text{A}(\triangle)$ other than $\text{A}(\triangle)$. If there is no such point, we set $x' = \hat{x}_\triangle$. See Figure 7(c). Note that $f_\triangle(x) \leq f_\triangle(x')$. Then we show that $f_\triangle(x') = d(\text{D}(\triangle), \text{A}(\triangle)) - \|\text{A}(\triangle) - x'\| \leq d(\text{D}(\triangle), x)$, which implies that $f_\triangle(x) \leq d(\text{D}(\triangle), x)$. If $d(\text{D}(\triangle), \text{A}(\triangle)) < \|\text{A}(\triangle) - x'\|$, the inequality is trivial since the left side of the inequality is negative. Thus let us assume that $d(\text{D}(\triangle), \text{A}(\triangle)) \geq \|\text{A}(\triangle) - x'\|$. Thus, there is a point x'' on $\pi(\text{D}(\triangle), \text{A}(\triangle))$ with $d(\text{A}(\triangle), x'') = \|\text{A}(\triangle) - x'\|$. In this case, $d(\text{D}(\triangle), x'') = d(\text{D}(\triangle), \text{A}(\triangle)) - \|\text{A}(\triangle) - x'\| \leq d(\text{D}(\triangle), x')$. Moreover, $\pi(\text{D}(\triangle), x)$ intersects the line segment connecting x' and $\text{A}(\triangle)$. The geodesic distance between the intersection and x' is less than the geodesic distance between the intersection and x. Thus, we have $d(\text{D}(\triangle), x') \leq d(\text{D}(\triangle), x)$. This implies that $f_\triangle(x') = d(\text{D}(\triangle), \text{A}(\triangle)) - \|\text{A}(\triangle) - x'\| \leq d(\text{D}(\triangle), x)$. Therefore, $f_\triangle(x) \leq d(\text{D}(\triangle), x)$ for any

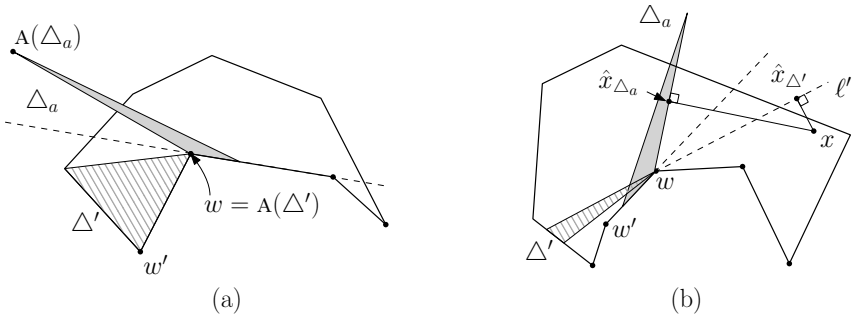

Figure 8 (a) rCell($\triangle'$) is not incident to rCell($\triangle_a$) since they lie in the different sides bounded by ℓ_b. (b) For x lying to the right of ℓ', $\|\hat{x}_{\triangle_a} - w\| < \|\hat{x}_{\triangle'} - w\|$, thus $f_{\triangle_a}(x) > f_{\triangle'}(x)$.

point $x \in T \cap G_{\text{Rtop}}(\triangle)$.

For a point $x \in G_{\text{in}}(\triangle)$, we have $f_{\triangle}(x) = d(\text{A}(\triangle), \text{D}(\triangle)) + \|\text{A}(\triangle) - x\| = d(\text{D}(\triangle), x)$. Thus, the lemma holds for any point in T. ◀

▶ **Lemma 22.** *Let $\triangle$ be an apexed triangle such that* rCell($\triangle$) $\cap\, T$ *is nonempty and $\pi(\text{D}(\triangle), \text{A}(\triangle))$ overlaps with the concave chain of T. Then $f_{\triangle}(x) \leq d(\text{D}(\triangle), x)$ for any point $x \in T \setminus G$, where G is the region contained in $G_{\text{Ltop}}(\triangle)$ bounded by ℓ_4 and the line containing ℓ_2. The equality holds if x lies in $\triangle$.*

Proof. All arguments in the proof of Lemma 21 hold, except for the case that $x \in G$. Note that the proof of Lemma 21 for the case that $x \in T \cap G_{\text{Rtop}}(\triangle)$ also holds for the case that $x \in T \cap (G_{\text{Ltop}}(\triangle) \setminus G)$. ◀

Now, we are ready to prove that the function we defined satisfies the conditions which the algorithm in [5] assumes. The following lemma implies that the abstract Voronoi diagram of the set A of apexed triangle $\triangle$ with rCell($\triangle$) $\cap\, \partial T \neq \phi$ restricted to T coincides with rFVD $\cap\, T$.

▶ **Lemma 23.** *For an apexed triangle $\triangle$ and a point $x \in T \cap$* rCell($\triangle$)*, x lies in* aCell($\triangle$)*.*

Proof. Assume to the contrary that there is an apexed triangle $\triangle$ and a point $x \in T \cap$ rCell($\triangle$) such that $x \notin$ aCell($\triangle$). This means that there is an apexed triangle $\triangle' \neq \triangle$ such that $f_{\triangle}(x) \leq f_{\triangle'}(x)$. Among all such apexed triangles, we choose the one with the maximum $f_{\triangle'}(x)$.

If $\pi(\text{D}(\triangle'), \text{A}(\triangle'))$ does not overlap with the concave chain of T, we have $f_{\triangle'}(x) \leq d(\text{D}(\triangle'), x)$ by Lemma 21. Note that x lies in $\triangle$ since $x \in$ rCell($\triangle$). By definition, we have $f_{\triangle'}(x) \leq d(\text{D}(\triangle'), x) < d(\text{D}(\triangle), x) = f_{\triangle}(x)$, which is a contradiction. If $\pi(\text{D}(\triangle'), \text{A}(\triangle'))$ overlaps with the concave chain of T and x lies in $T \setminus G$, where G is the region contained in $G_{\text{Ltop}}(\triangle)$ bounded by ℓ_4 and the line containing ℓ_2, we have $f_{\triangle'}(x) \leq d(\text{D}(\triangle'), x)$ by Lemma 22. Then this is a contradiction by an argument similar to the one for the case that $\pi(\text{D}(\triangle'), \text{A}(\triangle'))$ does not overlap with the concave chain of T.

The only remaining case is that $\pi(\text{D}(\triangle'), \text{A}(\triangle'))$ overlaps with the concave chain of T and $x \in T \cap G$. Let $w = \text{A}(\triangle')$. In the following, we show that there is an apexed triangle $\triangle_a$ such that $f_{\triangle'}(x) < f_{\triangle_a}(x)$. This is a contradiction as we chose the apexed triangle $\triangle'$ with maximum $f_{\triangle'}(x)$. We have $w \notin \text{int}(\text{rCell}(\triangle'))$. Since $\pi(\text{D}(\triangle'), \text{A}(\triangle'))$ overlaps with the concave chain of T, w lies in the concave chain of T. But it does not lie on the endpoints

of the chain since for every point $p \in \triangle'$, the combinatorial structure of $\pi(\mathrm{D}(\triangle'), p)$ is the same and $\pi(\mathrm{D}(\triangle'), p)$ contains $\mathrm{A}(\triangle')$.

If w lies on the boundary of $\mathsf{rCell}(\triangle')$, there is another apexed triangle $\triangle_a$ such that $f_{\triangle_a}(w) = f_{\triangle'}(w)$ and $w \in \triangle_a$. See Figure 8(a). This means that $\mathsf{rCell}(\triangle')$ and $\mathsf{rCell}(\triangle_a)$ share their boundaries. However, we have $\triangle' \cap \triangle_a = \{w\}$ since $\triangle'$ and $\triangle_a$ lie in the different sides of the line containing the bottom side of $\triangle_a$. Thus, no arc of rFVD is shared by $\mathsf{rCell}(\triangle')$ and $\mathsf{rCell}(\triangle_a)$, which is a contradiction.

If w does not lie on the boundary of $\mathsf{rCell}(\triangle')$, let w' be the vertex of T neighboring to w on the concave chain of T such that $d(\mathrm{D}(\triangle'), w) < d(\mathrm{D}(\triangle'), w')$. See Figure 8(b). In this case, there is an apexed triangle $\triangle_a \neq \triangle'$ such that $d(\mathrm{D}(\triangle_a), w) > d(\mathrm{D}(\triangle'), w)$ and the bottom side of $\triangle_a$ is contained in ww'. Let ℓ' be the line containing the side of $\triangle'$ other than its bottom side which is farther to w'. The point x lies in the side of ℓ' containing w'. Since $\triangle_a$ has its bottom side on the line containing ww', we have $\|w - \hat{x}_{\triangle_a}\| \leq \|w - \hat{x}_{\triangle'}\|$. Therefore, $f_{\triangle_a}(x) = d(\mathrm{D}(\triangle_a), w) - \|w - \hat{x}_{\triangle_a}\| \geq d(\mathrm{D}(\triangle_a), w) - \|w - \hat{x}_{\triangle'}\| > d(\mathrm{D}(\triangle'), w) - \|w - \hat{x}_{\triangle'}\| = d(\mathrm{D}(\triangle'), \mathrm{A}(\triangle')) - \|\mathrm{A}(\triangle') - \hat{x}_{\triangle'}\| = f_{\triangle'}(x)$ for all points $x \in T \cap G$, which is a contradiction.

Therefore, for all cases, the lemma holds. ◀

▶ **Corollary 24.** *The part of the abstract Voronoi diagram with the function $f_\triangle$ contained in T for all apexed triangles $\triangle \in A$ coincides with the refined farthest-point geodesic Voronoi diagram restricted to T.*

▶ **Lemma 25.** *For any two apexed triangles $\triangle_1$ and $\triangle_2$, the set $D(\triangle_1, \triangle_2)$ is connected.*

Proof. By Corollary 24, we have $\mathsf{rCell}(\triangle_1) \subseteq D(\triangle_1, \triangle_2)$. Recall that the bottom side of $\triangle_1$ is contained in $\mathsf{rCell}(\triangle_1)$ by the assumption we made at the beginning of this section. By Lemma 20, $G_{\mathrm{in}}(\triangle_1) \setminus \triangle_1$ is also contained in $D(\triangle_1, \triangle_2)$.

Let a be a point in $D(\triangle_1, \triangle_2) \setminus \mathsf{rCell}(\triangle_1)$. If $a \in G_{\mathrm{in}}(\triangle_1)$, the line segment ab connecting a and b is contained in $D(\triangle_1, \triangle_2)$ by Lemma 20, where b is the point on the bottom side of $\triangle_1$ at which the line passing through $\mathrm{A}(\triangle_1)$ and a crosses. Since b lies in $\mathsf{rCell}(\triangle_1)$, a lies in the connected component of $D(\triangle_1, \triangle_2)$ containing $\mathsf{rCell}(\triangle_1)$.

Thus we assume that $a \notin G_{\mathrm{in}}(\triangle_1)$. Note that $G_{\mathrm{in}}(\triangle_1)$ is a triangle. Let $b = \ell_1 \cap \partial B$ if a lies in $G_{\mathrm{Ltop}}(\triangle_1) \cup G_{\mathrm{Lside}}(\triangle_1)$, and $b = \ell_2 \cap \partial B$ otherwise. The point b lies in the connected component of $D(\triangle_1, \triangle_2)$ containing $\mathsf{rCell}(\triangle_1)$. Without loss of generality, we assume that ab is horizontal and a lies to the left of b. Then ab does not intersect the interior of $\triangle_1$. We claim that $ab \subseteq D(\triangle_1, \triangle_2)$, which implies that a lies in the connected component of $D(\triangle_1, \triangle_2)$ containing $\mathsf{rCell}(\triangle_1)$ and thus the lemma holds.

Consider $f_{\triangle_1}$ and $f_{\triangle_2}$ with domain ab. Since $ab \subset G_{\mathrm{Ltop}}(\triangle_1) \cup G_{\mathrm{Lside}}(\triangle_1)$ or $ab \subset G_{\mathrm{Rtop}}(\triangle_1) \cup G_{\mathrm{Rside}}(\triangle_1)$, the function $f_{\triangle_1}$ is linear on ab. Note that $f_{\triangle_1}(a) > f_{\triangle_2}(a)$ and $f_{\triangle_1}(b) > f_{\triangle_2}(b)$.

Let h_1 and h_3 denote the connected components of $ab \setminus \triangle_2$ containing a and b, respectively, and h_2 denote the set $ab \cap \triangle_2$. The functions $f_{\triangle_2}$ with domain h_1 and with domain h_3 are linear. On the other hand, $f_{\triangle_2}$ with domain h_2 is hyperbolic.

If $f_{\triangle_2}$ with domain h_1 and with domain h_3 are both increasing, the slope for $f_{\triangle_2}$ with domain h_3 is steeper than the slope for $f_{\triangle_2}$ with domain h_1. Similarly, if $f_{\triangle_2}$ with domain h_1 and with domain h_3 are both decreasing, the slope for $f_{\triangle_2}$ with domain h_1 is steeper than the slope for $f_{\triangle_2}$ with domain h_3. If one function is increasing and the other function is decreasing, the maximum for $f_{\triangle_2}$ with domain ab is either a or b since each contour curve consists of two line segments and at most one circular arc. Therefore, in any case, $f_{\triangle_2}(x) \leq f(x)$ for any point $x \in ab$, where $f : ab \to \mathbb{R}$ is the linear function with $f(a) = f_{\triangle_2}(a)$ and $f(b) = f_{\triangle_2}(b)$. This implies that $f_{\triangle_1}(x) > f_{\triangle_2}(x)$ for any point $x \in ab$. Therefore, $ab \subseteq D(\triangle_1, \triangle_2)$. ◀

With a similar argument, the following corollary also holds.

▶ **Corollary 26.** *For an apexed triangle $\triangle$ in any subset A' of A, $\bigcap_{\triangle' \neq \triangle \in A'} D(\triangle, \triangle')$ is nonempty and connected.*

▶ **Corollary 27.** *There exists at most one point $x \in B$ such that $f_{\triangle_1}(x) = f_{\triangle_2}(x) = f_{\triangle_3}(x)$ for any three apexed triangles $\triangle_1, \triangle_2$, and $\triangle_3$.*

▶ **Lemma 28.** *For any two apexed triangles $\triangle_1$ and $\triangle_2$ in A, the set $\{x \in B : f_{\triangle_1}(x) = f_{\triangle_2}(x)\}$ is a curve consisting of $O(1)$ algebraic curves. Moreover, both endpoints of the curve lie on ∂B.*

Proof. Consider contour curves one from $\triangle_1$ and the other from $\triangle_2$. At any fixed value, the contour curves for $\triangle_1$ and $\triangle_2$ cross in constant times under the assumption that no side of $\triangle_1$ is parallel to a side of $\triangle_2$. By this fact and Lemma 25, the set $\{x \in B : f_{\triangle_1}(x) = f_{\triangle_2}(x)\}$ is a curve whose endpoints lie on ∂B.

Consider the subdivision of B by overlaying the two subdivisions, one from $\triangle_1$ and the other from $\triangle_2$. There are at most nine cells in the subdivision of B. In each cell, $f_{\triangle_1}$ and $f_{\triangle_2}$ are algebraic functions. Thus, the set $\{x \in C : f_{\triangle_1}(x) = f_{\triangle_2}(x)\}$ is an algebraic curve for each cell C. ◀

▶ **Lemma 29.** *Let A be the set of all triangles with $\mathsf{rCell}(\triangle) \cap \partial T \neq \phi$ and A' be a subset of A. The abstract Voronoi diagram with $f_\triangle$ on B for all apexed triangles $\triangle$ in A forms a tree. Moreover, the abstract Voronoi diagram with $f_{\triangle'}$ on B for all triangles $\triangle'$ in A' forms a forest.*

Proof. To prove the first part of the lemma, it is sufficient to show that $\mathsf{aCell}(\triangle)$ is incident to ∂B for any apexed triangle $\triangle \in A$. Recall that $G_{\text{in}}(\triangle) \setminus \triangle$ is contained in $\mathsf{aCell}(\triangle)$. Moreover, $G_{\text{in}}(\triangle) \setminus \triangle$ intersects ∂B. Therefore, all $\mathsf{aCell}(\triangle)$ are incident to ∂B and the lemma holds.

Similarly, for the second part of the lemma, it is sufficient to show that the abstract Voronoi cell is incident to ∂B for any apexed triangle $\triangle \in A'$. This can be proved using a similar argument. ◀

Handling the case that some sides of two apexed triangles are parallel To handle the degenerate case, we modify $f_\triangle$ as follows by defining $\hat{x}_\triangle$ differently. First, we perturb ℓ_3 and ℓ_5 slightly such that ℓ_3 and ℓ_5 are circular arcs with common endpoint $\text{A}(\triangle)$ and the other endpoints on ∂B. We choose a sufficiently large number $r(\triangle)$ which is the common radius of ℓ_3 and ℓ_5. Then the center of ℓ_3 is the point on the ray from $\text{A}(\triangle)$ in the direction opposite to ℓ_1 with distance $r(\triangle)$ from $\text{A}(\triangle)$. Similarly, we choose the center of the circular arc ℓ_5. Then, for a point $x \in G_{\text{Lside}}(\triangle) \cup G_{\text{Ltop}}(\triangle)$, we map x into the point $\hat{x}_\triangle$ on the line containing ℓ_1 such that $r(\triangle) = \|\hat{x}_\triangle - c\| = \|x - c\|$, for some point c on the ray from $\text{A}(\triangle)$ in the direction opposite to ℓ_1. Similarly, we define $\hat{x}_\triangle$ for a point $x \in G_{\text{Rside}}(\triangle) \cup G_{\text{Rtop}}(\triangle)$. Now, each contour curve consists of three circular arcs.

There are two rules with regard to choosing $r(\triangle)$: (1) $r(\triangle) \neq r(\triangle')$ for any two distinct apexed triangles $\triangle$ and $\triangle'$, and (2) the apexed triangles $\triangle$ such that $\pi(\text{D}(\triangle), \text{A}(\triangle))$ overlaps with the concave chain of T use $r(\triangle)$ values greater than the ones for the other apexed triangles.

Let $f'_\triangle$ denote the new distance function. All lemmas and corollaries hold for $f'_\triangle$. They can be proved in a way similar to the proofs for the original distance function $f_\triangle$, except for Lemma 25. For Lemma 25, the proof of the lemma is valid only for a sufficiently large box

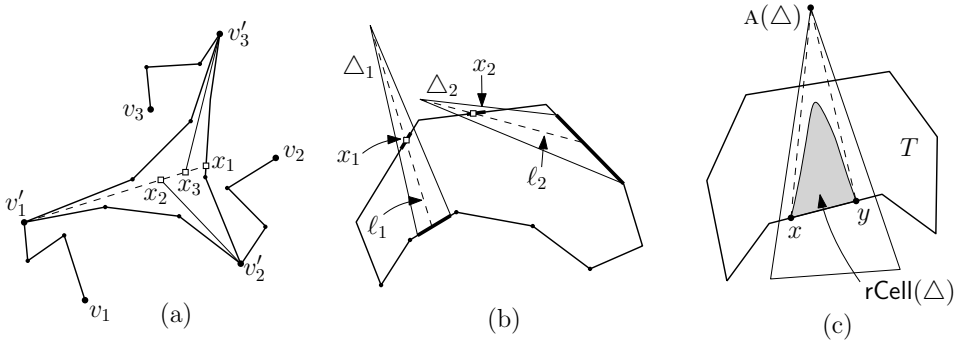

Figure 9 (a) The pseudo-triangle is subdivided into interior-disjoint four lune-cells. (b) The line segments ℓ_1 and ℓ_2 subdivide the lune-cell into three interior-disjoint lune-cells. Note that $\ell_1 \cap \ell_2 = \phi$. (c) We trim an apexed triangle $\triangle$ such that $\mathsf{rCell}(\triangle) \cap \partial T$ is connected.

B containing T. In this case, in $O(|A|)$ time, we can compute the size of a box which makes the proof valid. The size of such a box depends on the smallest $r(\triangle)$ for all apexed triangles $\triangle$ and the ratio of the slopes of two sides of any two apexed triangles whose refined cells are consecutive along ∂T.

Recall that the algorithm we described in this section assumes that (1) for any apexed triangle $\triangle$, $\mathsf{rCell}(\triangle) \cap \partial T$ is connected and contains the bottom side of $\triangle$, and (2) T is a lune-cell. In the following lemma, we show that we can compute $\mathsf{rFVD} \cap T$ for any base cell T using the algorithm we described.

▶ **Lemma 30.** *Given a base cell T from the subdivision algorithm, $\mathsf{rFVD} \cap T$ can be computed in $O(|\mathsf{rFVD} \cap \partial T| + |\partial T|)$ time.*

Proof. We subdivide each base cell further into subcells to satisfy the assumptions and apply the algorithm for each subcell. For a pseudo-triangle T, we subdivide the cell into four subcells as depicted in Figure 9(a). Let v_1, v_2, v_3 be three vertices of T such that the boundary of T consists of $\pi(v_1, v_2)$, $\pi(v_2, v_3)$ and $\pi(v_3, v_1)$, respectively.

Consider the three vertices v_1', v_2', v_3' of T such that the maximal common path for $\pi(v_i, v_j)$ and $\pi(v_i, v_k)$ is $\pi(v_i, v_i')$ for $i = 1, 2, 3$, where j and k are two distinct indices other than i.

First, we find a line segment $v_1'x_1 \subset T$ such that $v_1'x_1 \cap \partial T = \{v_1', x_1\}$. Then, we find two line segments $v_i'x_i \subset T$ such that $v_i'x_i \cap \partial T = \{v_i'\}$ and $x_i \in v_1'x_1$ for $i = 2, 3$. This can also be done in $O(|\mathsf{rFVD} \cap \partial T|)$ time. Then the three line segments $v_i'x_i$ subdivide T into four lune-cells T_1, T_2, T_3 and T_4 for $i = 1, 2, 3$. Note that to apply the algorithm in this section, $\mathsf{rFVD} \cap \partial T_j$ must be given. It can be computed in $O(|\mathsf{rFVD} \cap \partial T|)$ time by Corollary 17. Moreover, the total complexity of $\mathsf{rFVD} \cap \partial T_j$ for $j = 1, 2, 3, 4$ is $O(|\mathsf{rFVD} \cap \partial T|)$. Then we handle each lune-cell separately. Now every base cell is a lune-cell, thus satisfies the assumption (2).

We subdivide each lune-cell T_j further to satisfy the assumption (1). For an apexed triangle $\triangle$ appearing more than once, we subdivide T_j further as we did in the third phase of the subdivision by scanning the connected components of $\mathsf{rCell}(\triangle) \cap \partial T_j$. We find all connected components r' such that the line segment connecting r' and $\mathrm{A}(\triangle)$ intersects T_j other than its endpoints. Then we insert such connected components in the clockwise order along ∂C_j into a queue. Afterwards, we consider the connected components of $\mathsf{rCell}(\triangle) \cap \partial T_j$ which is not in the queue in the clockwise order along ∂C_j one by one.

To handle a connected component r, we do the following. We remove the first element

of the queue and denote it by r'. Let x be a point in the first element r' of the queue. If the line passing through x and $\text{A}(\triangle)$ intersects r, then we subdivide the cell by the line. We repeat this until the line passing through a point of the first element of the queue does not intersect r. Then, we handle the connected component next to r.

We do this for all apexed triangles such that $\mathsf{rCell}(\triangle)$ appears on ∂T_j more than once. Note that since $\mathsf{rCell}(\triangle)$'s are pairwise disjoint in their interiors, the line segments subdividing T_j do not intersect one another. Moreover, the extensions do not intersect rFVD. Thus we have $\mathsf{rFVD} \cap \partial T'_j$ for all subcells T'_j of T_j. Recall that rFVD is the set of the points which are not contained in any refined cell. Therefore, T_j is subdivided into $O(|\mathsf{rFVD} \cap \partial T_j|)$ subcells and the sum of $O(|\mathsf{rFVD} \cap \partial T'_j| + |\partial T'_j|)$ for all subcells T'_j is $O(|\mathsf{rFVD} \cap \partial T_j|)$.

Now, all apexed triangles appear on each cell at most once. If $\mathsf{rCell}(\triangle) \cap \partial T'_j$ contains a convex vertex v of T'_j (there are at most two such vertices), we decompose $\triangle$ into two triangles by the line passing through $\text{A}(\triangle)$ and v. Then only one of the triangles intersects the interior of T'_j by the definition of T'_j. We replace $\triangle$ with the triangle. Then, $\mathsf{rCell}(\triangle) \cap \partial T'_j$ is contained in an edge of T'_j for all apexed triangles $\triangle \in A$. Let x and y be the two endpoints of $\mathsf{rCell}(\triangle) \cap \partial T'_j$. See Figure 9(c). We reduce $\triangle$ into the triangle whose corners are x, y and $\text{A}(\triangle)$. From now on, when we refer an apexed triangle $\triangle$, we mean its trimmed triangle. Then $\mathsf{rCell}(\triangle) \cap T'_j$ is still contained in $\triangle$ by Lemma 4. We do this for all apexed triangles. Every apexed triangle with $\mathsf{rCell}(\triangle) \cap \partial T'_j \neq \phi$ has its bottom side on $\partial T'_j$, thus satisfies the assumption (1). Note that these subcells still satisfy the assumption (2).

Now, we apply the algorithm for each subcell, which can be done in time linear to the total complexity of $O(|\mathsf{rFVD} \cap \partial T'_j|)$ for all subcells T'_j from T, which is $O(|\mathsf{rFVD} \cap \partial T|)$. ◀

▶ **Theorem 31.** *The farthest-point geodesic Voronoi diagram of the vertices of a simple n-gon can be computed in $O(n \log\log n)$ time.*

6 A set of sites on the boundary

In this section, we show that the results presented above are general enough to work when the set S is an arbitrary set of sites contained in the boundary of P.

Since S is a subset of sites contained in ∂P, we can assume without loss of generality that all sites of S are vertices of P by splitting the edges where they lie on. In this section, we decompose the boundary of P into chains of consecutive vertices that share the same S-farthest neighbor and edges of P whose endpoints have distinct S-farthest neighbors. The following lemma is a counterpart of Lemma 1. Lemma 1 is the only place where it was assumed that S is the set of vertices of P.

▶ **Lemma 32.** *Given a set S of m sites contained in ∂P, we can compute the S-farthest neighbor of each vertex of P in $O(n + m)$ time.*

Proof. Let $w : P \to \mathbb{R}$ be a real valued function on the vertices of P such that for each vertex v of P,

$$w(v) = \begin{cases} D_P & \text{if } v \in S \\ 0 & \text{otherwise} \end{cases},$$

where D_P is any fixed constant larger than the diameter of P. Recall that the diameter of P can be computed in linear time [9].

For each vertex $p \in P$, we want to identify $\text{N}(p)$. To this end, we define a new distance function $d^* : P \times P \to \mathbb{R}$ such that for any two points u and v of P, $d^*(u, v) = d(u, v) + w(u) + w(v)$. Using a result from Hershberger and Suri [9, Section 6.1 and 6.3], in $O(n + m)$ time we can compute the farthest neighbor of each vertex of P with respect to d^*.

By the definition of the function w, the maximum distance from any vertex of P is achieved at a site of S. Therefore, the farthest neighbor from a vertex v of P with respect to d^* is indeed the S-farthest neighbor, $\text{N}(v)$, of v. ◀

▶ **Theorem 33.** *The farthest-point geodesic Voronoi diagram of m points on the boundary of a simple n-gon can be computed in $O((n+m)\log\log n)$ time.*

References

1 A. Aggarwal, L. J. Guibas, J. Saxe, and P. W. Shor. A linear-time algorithm for computing the Voronoi diagram of a convex polygon. *Discrete & Computational Geometry*, 4(6): 591–604, 1989.

2 H.-K. Ahn, L. Barba, P. Bose, J.-L. De Carufel, M. Korman, and E. Oh. A linear-time algorithm for the geodesic center of a simple polygon. In *Proceedings of the 31st Symposium on Compututaional Geometry, SoCG*, pages 209–223, 2015.

3 B. Aronov, S. Fortune, and G. Wilfong. The furthest-site geodesic Voronoi diagram. *Discrete & Computational Geometry*, 9(3):217–255, 1993.

4 T. Asano and G. Toussaint. Computing the geodesic center of a simple polygon. Technical Report SOCS-85.32, McGill University, 1985.

5 C. Bohler, R. Klein, and C. Liu. Forest-like abstract Voronoi diagrams in linear time. In *Proceedings of the 26th Canadian Conference on Computational Geometry, CCCG*, pages 133–141, 2014.

6 B. Chazelle. A theorem on polygon cutting with applications. In *Proceedings 23rd Annual Symposium on Foundations of Computer Science, FOCS*, pages 339–349, 1982.

7 H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 9(1):66–104, 1990.

8 L. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2(1):209–233, 1987.

9 J. Hershberger and S. Suri. Matrix searching with the shortest-path metric. *SIAM Journal on Computing*, 26(6):1612–1634, 1997.

10 J. S. B. Mitchell. Geometric shortest paths and network optimization. In *Handbook of Computational Geometry*, pages 633–701. Elsevier, 2000.

11 E. Papadopoulou. k-pairs non-crossing shortest paths in a simple polygon. *International Journal of Computational Geometry and Applications*, 9(6):533–552, 1999.

12 R. Pollack, M. Sharir, and G. Rote. Computing the geodesic center of a simple polygon. *Discrete & Computational Geometry*, 4(6):611–626, 1989.

13 S. Suri. Computing geodesic furthest neighbors in simple polygons. *Journal of Computer and System Sciences*, 39(2):220–235, 1989.