Computing Convex-Straight-Skeleton Voronoi Diagrams for Segments and Convex Polygons

Gill Barequet[1], Minati De[2](✉), and Michael T. Goodrich[3]

[1] Department of Computer Science, The Technion—Israel Institute of Technology, Haifa, Israel
barequet@cs.technion.ac.il

[2] Department of Mathematics, Indian Institute of Technology Delhi, New Delhi, India
minati@maths.iitd.ac.in

[3] Department of Computer Science, University of California, Irvine, Irvine, CA, USA
goodrich@uci.edu

Abstract. We provide efficient algorithms for computing compact representations of Voronoi diagrams using a convex-straight-skeleton (i.e., convex polygon offset) distance function when sites are line segments or convex polygons.

Keywords: Polygon-offset distance · Voronoi diagrams
Straight skeletons

1 Introduction

Voronoi diagrams (VD) are well-studied in a variety of fields, including, of course, computational geometry, but also ecology, biology, astro-physics, robot motion planning, and medical diagnosis (e.g., see [3,5]). Given a collection of disjoint geometric objects, such as points, segments, or polygons, which are called *sites*, a *Voronoi diagram* is a subdivision of the plane into *cells* such that all the points in a given cell have the same nearest site according to some distance metric.

There are many different types of distance functions that can be used to determine such nearest sites (e.g., see [5,15,19,20]), depending on the application, with one of particular interest for this paper being based on offsets of a convex polygon. Conceptually, this distance function is measured by locally translating the edges of an underlying convex polygon by some amount, either inwardly or outwardly. Such offset distance functions are motivated by applications in three-dimensional modeling and folding (e.g., see [1,7,8]) and are related to a structure known as the *straight skeleton* [2,10,13]. For this reason, we refer to such functions as *convex-straight-skeleton* distance functions.

M. De—Partially supported by DST-INSPIRE Faculty Grant (DST-IFA14-ENG-75).

L. Wang and D. Zhu (Eds.): COCOON 2018, LNCS 10976, pp. 130–142, 2018.
https://doi.org/10.1007/978-3-319-94776-1_12

Assuming that sites are line segments or convex polygons, the combinatorial complexities of convex-straight-skeleton Voronoi diagrams are given in a recent paper by Barequet and De [6], but they do not give efficient algorithms for computing such structures. Our interest in the present paper is the study of such efficient algorithms, for computing a compact representation of a convex-straight-skeleton Voronoi diagram for segments or convex polygons (where Voronoi edges comprising polygonal chains are represented implicitly).

1.1 Related Work

The Voronoi diagram of point sites is extensively studied in the literature (e.g., see [3,5]). Using the Euclidean metric, the combinatorial complexity for the Voronoi diagram is $O(n)$, where the sites are n points [12], or n disjoint line segments [4,18]. These diagrams can be constructed in $O(n \log n)$ time, which is worst-case optimal. For a set of n disjoint convex polygonal sites, each with complexity k, the Voronoi Diagram in the Euclidean metric for this set of sites has combinatorial complexity $O(kn)$ [14,16,20].

McAllister *et al.* [17] introduced the concept of a compact representation of a Voronoi diagram of convex polygonal sites, with distance defined either by the standard Euclidean metric or by scaling a convex polygon (which is related to but nevertheless different from the offset-polygon distance functions we study in this paper). They represent chains of piecewise-algebraic curves as single segments and they show that their compact representation can be used to quickly answer nearest-site queries and that, given a compact Voronoi diagram representation, one can compute the original Voronoi diagram in time proportionate to its combinatorial complexity. They show that such a compact Voronoi diagram can be constructed to have total size $O(n)$, where n is the total number of sites. They provide an algorithm running in time $O(n(\log n + \log k) \log m + m)$ for constructing such a compact Voronoi diagram of n convex polygons, each of size k, using a scaled distance function based on a convex m-gon.

Recently, Cheong *et al.* [11] showed that in the Euclidean metric, the farthest-site counterpart to the compact Voronoi diagram also has combinatorial complexity $O(n)$, and it can be computed in $O(n \log^3 n)$ time. On a related note, Bohler *et al.* [9] recently introduced the related notion of an abstract higher-order Voronoi diagram and studied its combinatorial complexity.

With respect to convex polygon-offset distance functions, the combinatorial complexity of the convex-straight-skeleton Voronoi diagram of a set of n point sites is shown by Barequet *et al.* [7] to be $O(nm)$, where m is the combinatorial complexity of the underlying convex polygon defining distance. Furthermore, they show that compact representations of such diagrams can be computed in $O(n(\log n + \log^2 m) + m)$ time. Recently, Barequet and De [6] show that the combinatorial complexity of a convex-straight-skeleton Voronoi diagram is $O(nm)$ for n line-segment sites and $O(n(m+k))$ for n convex polygons having at most k sides each. We are not aware of any previous results for efficient algorithms for computing a compact representation for a convex-straight-skeleton Voronoi diagram for line-segment or convex-polygon sites, however.

1.2 Our Contributions

In this paper, we show that it is possible to compute a compact representation of a convex-straight-skeleton Voronoi diagram of n line segments in $O(n(\log n + \log^2 m) + m^2)$ time and of n convex polygon sites, each of complexity at most k, in $O(n(\log n + \log k \log^2 m) + m^2)$ time.

Our algorithms are based on new insights into the geometry of convex-straight-skeleton distance functions with line segment and convex polygon sites, which allow us to show how to compute a number of geometric primitives efficiently for segments and convex polygons when the distance is defined by a convex offset-polygon distance function. For instance, we present an $O(\log m)$-time algorithm for computing the distance, $D_{\mathcal{P}}(z, s)$, between a point, z, and a line segment, s, using an offset distance defined by the polygon, $\mathcal{P}$. We also present an $O(\log^2 m)$-time algorithm for computing another elementary query, $vertex(s_1, s_2, s_3)$: Given three line segments s_1, s_2, and s_3, find the point which is equidistant from them. Our data structures for answering both of these types of queries require $O(m^2)$ preprocessing time. For convex polygon sites, we show that the elementary query operation, $D_{\mathcal{P}}(z, q)$, can be answered in $O(\log k \log m)$ time, where z is a point and q is a convex polygon with at most k sides. We also show that the primitive, $vertex(q_1, q_2, q_3)$, can be answered in $O(\log k \log^2 m)$ time, where z is a point and the q_i's are convex polygons with at most k sides. Both of these results use data structures having $O(m^2)$ preprocessing time.

2 Preliminaries

Let us borrow a few definitions from earlier papers [6,7]. Given a convex polygon, $\mathcal{P}$, described by the intersection of m closed half-planes, $\{H_i\}$, an *offset copy* of $\mathcal{P}$, denoted as $O_{\mathcal{P},\varepsilon}$, is defined as the intersection of the closed half-planes $\{H_i(\varepsilon)\}$, where $H_i(\varepsilon)$ is the half-plane parallel to H_i with bounding line translated by ε. Depending on whether the value of ε is positive or negative, the translation is respectively done outward or inward of P. See Fig. 1(a). Let $\varepsilon_0 < 0$ be the value for which $O_{\mathcal{P},\varepsilon_0}$ degenerates into a single point c (or a line segment s). We call the value, ε_0, the *negative radius* of $\mathcal{P}$, and the point c (or any point on s) the *center* of $\mathcal{P}$.

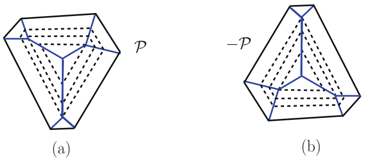

Fig. 1. (a) Offsets of a convex polygon, $\mathcal{P}$, along its straight skeleton (which is the same as its medial axis inside $\mathcal{P}$). (b) Offsets of the convex polygon ($-\mathcal{P}$), including its straight skeleton.

Using the above concept, the polygon-offset distance function $D_{\mathcal{P}}$ from one point to another point [7] and to an object [6] are defined as follows.

Definition 1 (Point to point distance [7]). *Let z_1 and z_2 be two points in $\mathbb{R}^2$ and $O_{\mathcal{P},\varepsilon}$ be an offset of $\mathcal{P}$ such that a translated copy of $O_{\mathcal{P},\varepsilon}$, centered at z_1, contains z_2 on its boundary. The offset distance is defined as* $D_{\mathcal{P}}(z_1, z_2) = \dfrac{\varepsilon + |\varepsilon_0|}{|\varepsilon_0|} = \dfrac{\varepsilon}{|\varepsilon_0|} + 1.$

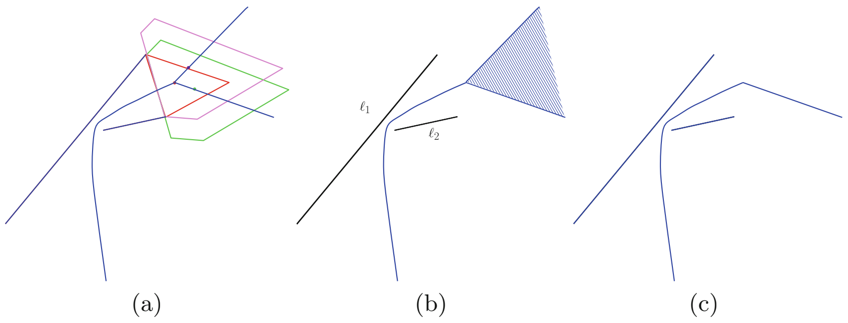

Fig. 2. (a) Three different positions of the offset polygons from where both line segments are equidistant; (b) The bisector (colored with blue) of two line segments according to the definition used in [7]; and (c) The bisector (colored with blue) according to our definition. (Color figure online)

Note that this distance function is not a metric since it is not symmetric. On the other hand, observe that $D_{\mathcal{P}}(z_1, z_2) = D_{(-\mathcal{P})}(z_2, z_1)$, where $(-\mathcal{P}) = \{-z | z \in \mathcal{P}\}$ is a "centrally-mirrored" copy of $\mathcal{P}$. See Fig. 1(b). This fact is widely used to compute the Voronoi diagram for point sites in [7].

Definition 2 (Point to object distance [6]**).** *Let z be any point, and let o be any object in $\mathbb{R}^2$. The offset distance $D_{\mathcal{P}}(z, o)$ is defined as $D_{\mathcal{P}}(z, o) = \min_{z' \in o} D_{\mathcal{P}}(z, z')$.*

2.1 Convex-Straight-Skeleton Voronoi Diagrams

Under the convex polygon offset distance function, the bisector of two points (as defined originally [7]) can be 2-dimensional instead of 1-dimensional (see Fig. 2 for an illustration). This makes the Voronoi diagram of points unnecessarily complicated. To make it simple, as is also done by Klein and Woods [15], the bisector and Voronoi diagram with respect to the offset distance function $D_{\mathcal{P}}$ is defined as follows [6].

Let z be a point, and $\Sigma = \{\sigma_i\}$ a set of objects in the plane. In order to avoid 2-dimensional bisectors between two objects in Σ, we define the index of the objects as the "tie breaker" for the relation '$\prec$' between distances from z to the sites. That is, $D_{\mathcal{P}}(z, \sigma_i) \prec D_{\mathcal{P}}(z, \sigma_j)$, if $D_{\mathcal{P}}(z, \sigma_i) < D_{\mathcal{P}}(z, \sigma_j)$ or, in case $D_{\mathcal{P}}(z, \sigma_i) = D_{\mathcal{P}}(z, \sigma_j)$, if $i < j$. Note that the relation '$\prec$' does not allow equality if $i \neq j$. Therefore, the definition below uses the *closure* of portions of the plane in order to have proper boundaries between the regions of the diagram.

Definition 3 (Convex-Straight-Skeleton Voronoi diagram). *Let $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ be a set of n sites in $\mathbb{R}^2$. For any $\sigma_i, \sigma_j \in \Sigma$, we define the region of σ_i with respect to σ_j as $NV_{\mathcal{P}}^{\sigma_j}(\sigma_i) = \{z \in \mathbb{R}^2 | D_{\mathcal{P}}(z, \sigma_i) \prec D_{\mathcal{P}}(z, \sigma_j)\}$. The bisecting curve $B_{\mathcal{P}}(\sigma_i, \sigma_j)$ is defined as $\overline{NV_{\mathcal{P}}^{\sigma_j}(\sigma_i)} \cap \overline{NV_{\mathcal{P}}^{\sigma_i}(\sigma_j)}$, where $\overline{X}$ is* *the closure of X. The region of a site σ_i in the convex straight-skeleton Voronoi diagram of Σ is defined as*

$$NV_{\mathcal{P}}(\sigma_i) = \{z \in \mathbb{R}^2 \mid D_{\mathcal{P}}(z, \sigma_i) \prec D_{\mathcal{P}}(z, \sigma_j) \quad \forall j \neq i\}.$$

The nearest-site convex straight-skeleton Voronoi diagram is the union of the regions

$$NVD_{\mathcal{P}}(\Sigma) = \bigcup_i NV_{\mathcal{P}}(\sigma_i).$$

In other words, the diagram $\mathrm{NVD}_{\mathcal{P}}(\Sigma)$ is a partition of the plane, such that if a point $p \in \mathbb{R}^2$ has more than one closest site, then it belongs to the region of the site with the smallest index. The bisectors between regions are defined by taking the closures of the open regions.

3 Tools for Constructing Convex Straight-Skeleton Voronoi Diagrams

Let us generalize our distance function to object-to-point distance as follows.

Definition 4 (Object-to-point distance). *Let z be any point, and let o be any object in $\mathbb{R}^2$. The offset distance $D_{\mathcal{P}}(o, z)$ is defined as $D_{\mathcal{P}}(o, z) = \min_{z' \in o} D_{\mathcal{P}}(z', z)$.*

The following lemma is crucial for the correctness of our algorithms.

Lemma 5. *$D_{\mathcal{P}}(z, o) = D_{(-\mathcal{P})}(o, z)$.*

3.1 Tools for Line Segments

In [7], the strong relationship between the continuous change of $O_{(-\mathcal{P}),\varepsilon}$ (as a function of ε) and the medial axis of $(-\mathcal{P})$ was observed. The *medial axis*, which is also the *straight skeleton* of the convex polygon $(-\mathcal{P})$ is defined as the set of points inside $(-\mathcal{P})$ that have more than one closest point among the points of $\partial(-\mathcal{P})$. It was noticed that if we change the value of ε continuously by fixing the center, then the vertices of $O_{(-\mathcal{P}),\varepsilon}$ slide along the edges of the medial axis. Outside the polygon, the medial axis and straight skeleton differ in that the straight skeleton extends outward as bisectors of edge offsets, whereas the medial axis extends "rounded"

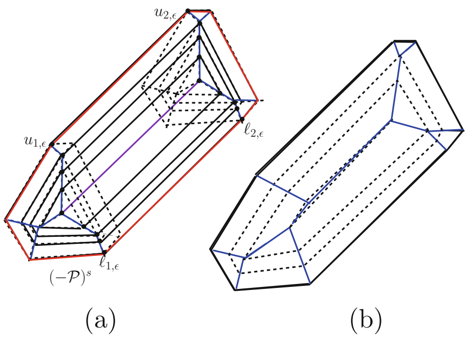

Fig. 3. (a) A convex polygon $(-\mathcal{P})^s$ (marked with red), and $O_{(-\mathcal{P})^s,\varepsilon}$ for different values of ε; the extruded medial axis of $(-\mathcal{P})^s$ is marked with blue; and (b) The medial axis of $(-\mathcal{P})^s$ (marked with blue). (Color figure online)

edges from vertices. This information was widely used to efficiently compute $D_{\mathcal{P}}(z_1, z_2)$ for any two points z_1 and z_2 in $\mathbb{R}^2$. As a result, the preprocessing of the medial axis of $(-\mathcal{P})$ in a tree-like data structure was sufficient to answer $D_{\mathcal{P}}(z_1, z_2)$ queries in $O(\log m)$ time, for any two points z_1 and z_2 in $\mathbb{R}^2$.

Lemma 6 *[7, Theorem 10]. Allowing $O(m)$ time preprocessing of the polygon $(-\mathcal{P})$, the distance function $D_{\mathcal{P}}(z_1, z_2)$ can be computed in $O(\log m)$ time, where z_1 and z_2 are two points.*

For our purposes, we would like to preprocess the underlying polygon and compute a similar data structure which will enable us to answer $D_{\mathcal{P}}(z, s)$ queries efficiently, for any point z and line segment s in $\mathbb{R}^2$.

Let $(-\mathcal{P})^s$ be the convex polygon obtained by taking the union of all the translated copies of $(-\mathcal{P})$ centered at all the points in the line segment s. Similarly, we define $O_{(-\mathcal{P})^s,\varepsilon}$ as the convex polygon obtained by taking the union of all the translated copies of $O_{(-\mathcal{P}),\varepsilon}$ centered at all the points in the line segment s. Note that $(-\mathcal{P})^s$ (resp., $O_{(-\mathcal{P})^s,\varepsilon}$) is the convex-hull of two translated copies of $(-\mathcal{P})$ (resp., $O_{(-\mathcal{P}),\varepsilon}$) centered at the two endpoints of s. Note that when ε takes the value ε_0, which is the negative radius of $(-\mathcal{P})$, then $O_{(-\mathcal{P})^s,\varepsilon_0}$ degenerates into the line segment s. We refer to the line segment s as the *center* of $O_{(-\mathcal{P})^s,\varepsilon}$, for $\varepsilon \geq \varepsilon_0$. Note that even in the worst case, the complexity of $(-\mathcal{P})^s$ is not twice the complexity of $\mathcal{P}$, but simply $|(-\mathcal{P})^s| = |\mathcal{P}| + 2$. We define the *extruded medial axis* of $(-\mathcal{P})^s$ as the set of points inside $(-\mathcal{P})^s$ such that if we change the value of ε continuously by fixing the center at s, then the vertices of $O_{(-\mathcal{P})^s,\varepsilon}$ slide along the edges of the extruded medial axis (see Fig. 3(a)). Also note that the extruded medial axis of $(-\mathcal{P})^s$ may not be the same as the medial axis of $(-\mathcal{P})^s$ (see Figs. 3(a–b) for a comparison).

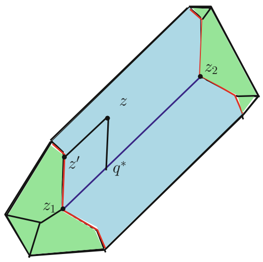

Fig. 4. The *extruded medial axis* of $(-\mathcal{P})^s$: The medial region and parallel region are colored with light green and light blue, respectively; two common boundaries are marked with red. (Color figure online)

We define the following distance function.

Definition 7 **($D_{(-\mathcal{P})^s}(s, z)$).** *Let z be any point and s be any line segment in $\mathbb{R}^2$, and $O_{(-\mathcal{P})^s,\varepsilon}$ be an offset of $(-\mathcal{P})^s$ (centered at s) such that $O_{(-\mathcal{P})^s,\varepsilon}$ contains z on its boundary. The offset distance $D_{(-\mathcal{P})^s}(s, z)$ is defined as* $D_{(-\mathcal{P})^s}(s, z) = \dfrac{\varepsilon + |\varepsilon_0|}{|\varepsilon_0|} = \dfrac{\varepsilon}{|\varepsilon_0|} + 1.$

Lemma 8. $D_{(-\mathcal{P})}(s, z) = D_{(-\mathcal{P})^s}(s, z)$.

Combining Lemmata 5 and 8, we have the following.

Lemma 9. $D_{\mathcal{P}}(z, s) = D_{(-\mathcal{P})^s}(s, z)$.

Let us now show how to compute efficiently the distance $D_{\mathcal{P}}(z, s)$ for any point z and line segment s in $\mathbb{R}^2$. Following Lemma 9, we know that it is sufficient to compute $D_{(-\mathcal{P})^s}(s, z)$. Provided that the extruded medial axis of $(-\mathcal{P})^s$ is computed in a preprocessing step, $D_{(-\mathcal{P})^s}(s, z)$ can be computed as in Lemma 6 [7]. Note that $D_{\mathcal{P}}(z, s)$ is a primitive operation for the computation of the compact Voronoi diagram. Thus, simply preprocessing the extruded medial axis of $(-\mathcal{P})^s$ for every segment $s \in \mathscr{S}$ would result in increased preprocessing space (i.e., $O(nm)$ space). However, we prove here that even with preprocessing only the medial axis of $(-\mathcal{P})$, we can compute $D_{\mathcal{P}}(z, s)$ with the same query time as that of computing $D_{\mathcal{P}}(z_j, z_\ell)$, $z_j, z_\ell \in \mathbb{R}^2$.

Let us now illustrate the properties of the extruded medial axis of $(-\mathcal{P})^s$. Place two translated copies $T_{1,\varepsilon}$ and $T_{2,\varepsilon}$ of the offset polygon $O_{(-\mathcal{P}),\varepsilon}$, centered at two endpoints z_1 and z_2 of the line segment s (see Fig. 3(a)). Let $u_{1,\varepsilon}$ and $u_{2,\varepsilon}$ be the two vertices of the upper tangent[1] of the convex-hull joining $T_{1,\varepsilon}$ and $T_{2,\varepsilon}$. Observe that $u_{1,\varepsilon}$ and $u_{2,\varepsilon}$ are the same vertex of the offset polygon $O_{(-\mathcal{P}),\varepsilon}$. Similarly, let $\ell_{1,\varepsilon}$ and $\ell_{2,\varepsilon}$ be the two vertices of the lower tangent of the convex-hull joining $T_{1,\varepsilon}$ and $T_{2,\varepsilon}$. Note that if we change the value of ε, then both $u_{i,\varepsilon}$ and $\ell_{i,\varepsilon}$ change along the medial axis of $T_{i,\varepsilon}$, $i \in \{1, 2\}$, and the upper (resp., lower) tangent moves and is always parallel to s. Thus, the extruded medial axis of $(-\mathcal{P})^s$ is a subset of the union of the medial axes of $T_{1,\varepsilon}$ and $T_{2,\varepsilon}$. Specifically, the portion of the medial axis of $T_{i,\varepsilon}$, that lies between $u_{i,\varepsilon}$ and $\ell_{i,\varepsilon}$ and whose end vertices are in the convex-hull, are in the extruded medial axis of $(-\mathcal{P})^s$. We refer to this part of the medial axis of $T_{i,\varepsilon}$ as the *medial region* with respect to the line segment s. We define the *parallel region* as the part of the polygon $(-\mathcal{P})^s$ that does not have dominating edges from the medial region (see Fig. 4 for an illustration). For a point z which belongs to the parallel region of $(-\mathcal{P})^s$, let z' be the projection of z on the *common boundary* of the medial region of T_i, $i \in \{1, 2\}$ and the parallel region (see Fig. 4). Then, $D_{\mathcal{P}}(z, s)$ can be calculated by simply computing $D_{\mathcal{P}}(z', z_i)$.

Lemma 10. *Allowing $O(m^2)$ time preprocessing of the polygon $(-\mathcal{P})$, $D_{\mathcal{P}}(z, s)$ can be computed in $O(\log m)$ time, where z is a point and s is a line segment. Along with that, a point $q^* \in s$ satisfying $D_{\mathcal{P}}(z, q^*) = D_{\mathcal{P}}(z, s)$ can be reported in the same amount of time.*

[1] If s is vertical, then we arbitrarily choose the left tangent as the upper.

Proof. We keep two copies, T_1 and T_2, of the processed medial axis of $(-\mathcal{P})$ as required by Lemma 6. In addition, we preprocess $(-\mathcal{P})$ such that both traversing and binary searching is possible along any vertex-to-center path of $(-\mathcal{P})$. Since there are m vertices, there are m such paths. By simply storing each path as a list, we need a total of $O(m^2)$ space and time to preprocess this data structure[2].

To answer the query $D_{\mathcal{P}}(z, s)$, we do the following:

Step 1. Compute the upper and lower tangents of the two translated copies T_1 and T_2, centered at z_1 and z_2, respectively, where z_1 and z_2 are the endpoints of the line segment s. Let u_i and ℓ_i be the points in which the upper and lower tangents T_i, $i \in \{1, 2\}$, touch the two polygons.

Step 2.1. If z is in the relevant region of T_i with respect to s, then answer $D_{\mathcal{P}}(z, s)$, return the point q^* by evoking $D_{\mathcal{P}}(z, z_i)$, and stop.

Step 2.2. Otherwise (if z is in the parallel region), we answer a ray-shooting query on the common boundary B_i between the medial region of T_i and the parallel region of $(-\mathcal{P})^s$ to find z', the projection of z on B_i (see Fig. 4). We obtain $D_{\mathcal{P}}(z, s)$ by computing $D_{\mathcal{P}}(z', z_i)$. We can then find the point q^* by translating the point z_i by $d(z, z')$ along the line segment s, where $d(z, z')$ is the Euclidean distance between z and z'.

Step 1 takes $O(\log m)$ time, assuming that $(-\mathcal{P})$ is available as a cyclic list of vertices. Step 2.1 can be done in $O(\log m)$ time as in Lemma 6. Since B_i is a path from a vertex of $(-\mathcal{P})$ to its center along the medial axis where we can perform a binary search, we can find z' in $O(\log m)$ time. Hence, Step 2.2 takes $O(\log m)$ time. In total, the query time complexity is $O(\log m)$. $\square$

Another primitive operation is finding the Voronoi vertex v^* where the Voronoi cells with respect to the polygon-offset distance $D_{\mathcal{P}}$ for sites s_1, s_2, s_3 occur in counterclockwise order around v^*. Applying the tentative prune-and-search paradigm, we can find v^* similarly to the method in [7]. The only difference is that here, three different polygons $(-\mathcal{P})^{s_1}$, $(-\mathcal{P})^{s_2}$, and $(-\mathcal{P})^{s_3}$ come into the picture instead of three identical copies of $(-\mathcal{P})$. Thus, we have the following:

Lemma 11. *Given three line segments s_1, s_2, s_3 in the plane, and a convex polygon $\mathcal{P}$ of m sides, a Voronoi vertex v^*, where the Voronoi cells with respect to the polygon-offset distance $D_{\mathcal{P}}$ for sites s_1, s_2, s_3 occur in counterclockwise order around v^*, can be computed in $O(\log^2 m)$ time (allowing $O(m^2)$ time for preprocessing).*

[2] This preprocessing step can probably be implemented in a more efficient way, but since it's not the bottleneck of the algorithm, such an improvement will not affect the total running time of the algorithm for computing $D_{\mathcal{P}}(z, s)$.

3.2 Tools for Convex Polygonal Sites

Here, we generalize the formerly described tool of line segments for convex polygons. Let q be a convex polygon with k sides, and ∂q be the boundary of q. Let $(-\mathcal{P})^q$ be the convex polygon obtained by uniting all the translated copies of $(-\mathcal{P})$ centered at all the points $z \in q$. Similarly, we define an offset copy $O_{(-\mathcal{P})^q,\varepsilon}$ as the convex polygon obtained by uniting all the translated copies of $O_{(-\mathcal{P}),\varepsilon}$ centered at all the points $z \in q$. Note that $(-\mathcal{P})^q$ (resp., $O_{(-\mathcal{P})^q,\varepsilon}$) is the convex-hull of k translated copies of $(-\mathcal{P})$ (resp., $O_{(-\mathcal{P}),\varepsilon}$) centered at the k vertices of the polygon q. Note that when ε takes the value ε_0, which is the negative radius of $(-\mathcal{P})$, then $O_{(-\mathcal{P})^q,\varepsilon_0}$ degenerates into the polygon q. We refer to the polygon q as the *center* of $O_{(-\mathcal{P})^q,\varepsilon}$, for $\varepsilon \geq \varepsilon_0$. The complexity of $(-\mathcal{P})^q$ is $|(-\mathcal{P})^q| = |\mathcal{P}| + k$ because each side of $(-\mathcal{P})$ can appear at most once along the boundary of $(-\mathcal{P})^q$, and there are exactly k tangents. We define the *extruded medial axis* of $(-\mathcal{P})^q$ as the set of points inside $(-\mathcal{P})^q$ such that if we change the value of $\varepsilon \geq \varepsilon_0$ continuously by fixing the center at q, then the vertices of $O_{(-\mathcal{P})^q,\varepsilon}$ slide along the edges of the extruded medial axis (see Fig. 5). Similarly to the previous section, the extruded medial axis of $(-\mathcal{P})^q$ may differ from the medial axis of $(-\mathcal{P})^q$.

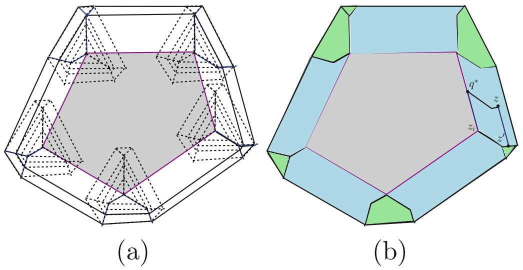

Fig. 5. (a) $(-\mathcal{P})^q$: The polygon q is colored gray. Translated copies of $(-\mathcal{P})$, centered at the vertices of q, are shown with dotted lines. The extruded medial axis of $(-\mathcal{P})^q$ is marked with blue. (b) The medial region and parallel region are colored light green and light blue, respectively. (Color figure online)

We define the following distance function.

Definition 12 ($D_{(-\mathcal{P})^q}(q,z)$). *Let z be any point and q be any convex polygon with k sides in $\mathbb{R}^2$, and $O_{(-\mathcal{P})^q,\varepsilon}$ be an offset copy of $(-\mathcal{P})^q$ (centered at q) such that $O_{(-\mathcal{P})^q,\varepsilon}$ contains z on its boundary. The offset distance $D_{(-\mathcal{P})^q}(q,z)$ is defined as* $D_{(-\mathcal{P})^q}(q,z) = \dfrac{\varepsilon + |\varepsilon_0|}{|\varepsilon_0|} = \dfrac{\varepsilon}{|\varepsilon_0|} + 1.$

Lemma 13. *$D_{(-\mathcal{P})}(q,z) = D_{(-\mathcal{P})^q}(q,z)$.*

Proof. The proof is similar to the proof of Lemma 8. □

Combining Lemmata 5 and 13, we have the following.

Lemma 14. *$D_{\mathcal{P}}(z,q) = D_{(-\mathcal{P})^q}(q,z)$.*

Let us illustrate the properties of the extruded medial axis of $(-\mathcal{P})^q$. Place k translated copies $T_{i,\varepsilon}$, $i \in \{1,\ldots,k\}$, of the offset polygon $O_{(-\mathcal{P}),\varepsilon}$ centered at the k vertices $\{z_i\}$, $i \in \{1, \dots, k\}$, of the polygon q (see Fig. 5(a)). Here $T_{i,\varepsilon}$ and $T_{i+1,\varepsilon}$ are two clockwise consecutive copies, and let t_i be the outer common tangent of $T_{i,\varepsilon}$ and $T_{i+1,\varepsilon}$, where the addition of subtraction of the index i are modulo k. Let $u_{i_1,\varepsilon}$ and $u_{i_2,\varepsilon}$ be the two vertices of the outer tangent of the convex-hull joining $T_{i,\varepsilon}$ and $T_{i+1,\varepsilon}$. Observe that $u_{i_1,\varepsilon}$ and $u_{i_2,\varepsilon}$ are the same vertex of the offset polygon $O_{(-\mathcal{P}),\varepsilon}$. Note that if we change the value of ε, then all vertices $u_{i_j,\varepsilon}$ move along the medial axis of $T_{i,\varepsilon}$, $i \in \{1, 2, \dots, k\}$, $j \in \{1, 2\}$, and all k outer tangents move parallel to themselves. Thus, the extruded medial axis of $(-\mathcal{P})^q$ is a subset of the union of medial axes of $T_{i,\varepsilon}$, $i \in \{1, 2, \dots, k\}$ (see Fig. 5(b)). Specifically, the portion of the medial axis of $T_{i,\varepsilon}$ that lies between $u_{(i-1)_2,\varepsilon}$ and $u_{i_1,\varepsilon}$ and whose end vertices appear in $O_{(-\mathcal{P})^q,\varepsilon}$ are in the extruded medial axis of $(-\mathcal{P})^q$. We refer to this part of the medial axis of $T_{i,\varepsilon}$ as the *medial region* of $T_{i,\varepsilon}$ with respect to the polygon q. We define the ith *parallel region* as the part of the polygon $O_{(-\mathcal{P})^q,\varepsilon}$ that does not have dominating edges from the medial region and lies between $T_{i,\varepsilon}$ and $T_{i+1,\varepsilon}$ (see Fig. 5(b)). For a point z which belongs to the ith parallel region of $O_{(-\mathcal{P})^q,\varepsilon}$, let z' be the projection of z on the *common boundary* of the medial region of T_i and the ith parallel region. Then, $D_{\mathcal{P}}(z, q)$ can be found by simply computing $D_{\mathcal{P}}(z', z_i)$.

Lemma 15. *Allowing $O(m^2)$ time and space for preprocessing of the polygon $\mathcal{P}$, the function $D_{\mathcal{P}}(z, q)$ can be computed in $O(\log k \log m)$ time, where z is a point, and q is a convex polygon with at most k sides.*

Proof. We keep three copies T_j, $j \in \{-1, 0, 1\}$ of the processed medial axis of $(-\mathcal{P})$ as required by Lemma 6. In addition, we preprocess $(-\mathcal{P})$ such that both traversing and binary searching are possible along each vertex-to-center path of $(-\mathcal{P})$. Since there are m vertices, there are m such paths. By simply storing each path as a list, we need a total of $O(m^2)$ space and time to preprocess this data structure (See footnote 2).

To answer the query $D_{\mathcal{P}}(z, s)$, we perform a binary search along the cyclic list of vertices of the polygon q. At each step of the binary search, we select, say, z_i, the ith vertex of q, and decide whether either (i) z is in the medial region; (ii) z is in the parallel region of the corresponding translated copy of $(-\mathcal{P})$; or (iii) z is in the left or right side of z_i in the cyclic order list of vertices of q.

To decide whether z is in the medial region or in the parallel region of the corresponding translated copy of $(-\mathcal{P})$, centered at the ith vertex, we do the following:

Step 1. Place three copies T_j, $j \in \{-1, 0, 1\}$ at the $(i-1)$st, ith, and $(i+1)$st vertices of q. Let z_i be the ith vertex of q.

Step 2. Find the outer common tangents of T_{-1}, T_0 and of T_0, T_1. This will allow us to detect the medial region and parallel region of T_0 with respect to q.

Step 3.1. If z is in the medial region of T_0, then we can answer $D_{\mathcal{P}}(z, q)$ and report the point q^* by invoking $D_{\mathcal{P}}(z, z_i)$. We stop after reporting.

Step 3.2. Else, if z is in the parallel region (see Fig. 5(b)), we perform ray-shooting on the common boundary B_i between the medial region and the parallel region of T_i to find z', the projection of z on B_i. We obtain $D_{\mathcal{P}}(z,s)$ by computing $D_{\mathcal{P}}(z',z_i)$. We can find the point q^* by translating the point z_i by $d(z,z')$ along the line segment s, where $d(z,z')$ is the Euclidean distance between z and z'.

Step 3.3. Otherwise, determine the side where z lies with respect to z_i in the cyclic list of vertices of q.

Step 1 takes constant time, Step 2 takes $O(\log m)$ time assuming that the polygon $(-\mathcal{P})$ is available as a cyclic list of vertices. Step 3.1 can be done in $O(\log m)$ time as in Lemma 6. Since B_i is a path from a vertex of $(-\mathcal{P})$ to its center along the medial axis where we can do binary search, we can find z' in $O(\log m)$ time. Therefore, Step 3.2 takes $O(\log m)$ time. Step 3.3 needs constant time. Thus, at each step of the binary search, we need $O(\log m)$ time. In total, the query time complexity is $O(\log k \log m)$. □

The other primitive operation is to find the Voronoi vertex v^*, where the Voronoi cells, with respect to the polygon-offset distance $D_{\mathcal{P}}$ for three polygonal sites q_1, q_2, q_3, occur in counterclockwise order around v^*. Applying the tentative prune-and-search paradigm, we can find v^* similarly to the method used in [7]. The main difference is that here, three different polygons $(-\mathcal{P})^{q_1}$, $(-\mathcal{P})^{q_2}$ and $(-\mathcal{P})^{q_3}$ come into the picture instead of three identical copies of $(-\mathcal{P})$. On the other hand, each $O(\log m)$-time distance evaluation function is replaced by an $O(\log k \log m)$-time operation for evaluating $D_{\mathcal{P}}(z,q)$ (by Lemma 15). Thus, we have the following:

Lemma 16. *Given three polygons p_1, p_2, p_3 in the plane, each having at most k sides, and a convex polygon $\mathcal{P}$ with m sides, the point v^*, equidistant from p_1, p_2, p_3 with respect to the polygon-offset distance $D_{\mathcal{P}}$, can be computed in $O(\log k \log^2 m)$ time (allowing $O(m^2)$ time for preprocessing).*

4 Computing a Compact Convex Straight-Skeleton Voronoi Diagram

As mentioned above, McAllister *et al.* [17] presented an algorithm for computing a *compact* representation of the nearest-site Voronoi diagram of a set of convex polygonal sites with respect to a convex (scaled) distance function. Here, we show how to adapt their method to obtain a compact representation of $NVD_{\mathcal{P}}(\mathscr{Q})$, where $\mathscr{Q} = \{q_1, q_2, \ldots, q_n\}$ is a set of n convex polygonal sites, each having at most k sides, and $\mathrm{NVD}_{\mathcal{P}}(\mathscr{Q})$ is the nearest-site convex-straight-skeleton Voronoi diagram of these sites, with respect to the convex polygon-offset distance function $D_{\mathcal{P}}$, where $\mathcal{P}$ is an m-sided convex polygon.

For any point z and a polygonal site q, *spoke*(z,q) is defined as a line segment $\overline{z, z^*}$, such that $D_{\mathcal{P}}(z,z^*) = \min\limits_{z' \in q} D_{\mathcal{P}}(z,z')$. Here, z^* is referred to as the *attachment point* of the spoke. Note that *spoke*(z,q) can be computed in $O(\log k \log m)$ time (with $O(m^2)$ preprocessing time), where the polygon q has k vertices/edges (Lemma 15).

For three sites q_1, q_2, q_3, $vertex(q_1, q_2, q_3)$ is defined as the point v equidistant from q_1, q_2, q_3 with respect to the polygon-offset distance function $D_{\mathcal{P}}$. From Lemma 16 we know that $vertex(q_1, q_2, q_3)$ can be computed in $O(\log k \log^2 m)$ time (with $O(m^2)$ preprocessing time).

The compact Voronoi diagram is a simplified version of the full Voronoi diagram. Here, we maintain a set of spokes from the Voronoi vertices around the cell, and each polygonal site q is replaced by its *core*, where a core is the convex hull of the attachment points that lie on the boundary of q (see [17, Fig. 4]). As a result, we obtain a compact representation whose combinatorial complexity is $O(n)$. Note that the combinatorial complexity of $NVD_{\mathcal{P}}(\mathscr{Q})$ is $O(n(m+k))$ [6], which is higher than the combinatorial complexity of the compact representation.

Note that each cell of this compact diagram is actually composed of portions of two cells of the full Voronoi diagram. Thus, we can do the point location as follows. Given a point z, we can obtain the compact cell and the two corresponding candidate sites q_i and q_j in $O(\log n)$ time. Then, spending additional $O(\log k \log m)$ time to compare $D_{\mathcal{P}}(z, q_i)$ and $D_{\mathcal{P}}(z, q_j)$, we can determine the identity of the cell of the full Voronoi diagram in which the point is located.

As observed in [7, Sect. 4], the geometric properties of the compact Voronoi diagram are preserved when we use a convex polygon-offset distance function instead of a convex distance function. Hence, we can apply Theorem 3.10 of [17], which states that the compact representation of the Voronoi diagram can be computed in $O(n(\log n + T_v))$ time, where T_v is the time needed for performing primitive operations like $spoke(z, q)$ and $vertex(q_1, q_2, q_3)$. For convex polygonal sites, T_v is $O(\log k \log^2 m)$ (this follows from Lemmata 15 and 16). Thus, we can compute a compact Voronoi diagram for $NVD_{\mathcal{P}}(\mathscr{Q})$ in $O(n(\log n + \log k \log^2 m) + m^2)$ time, where $\mathscr{Q}$ is a set of n disjoint convex polygonal sites, each having at most k sides.

Theorem 17. *For a set $\mathscr{Q}$ of n convex polygonal sites, each having at most k sides, the compact representation of the Voronoi diagram $NVD_{\mathcal{P}}(\mathscr{Q})$ can be computed in expected $O(n(\log n + \log k \log^2 m) + m^2)$ time, where m is the number of sides of the underlying convex polygon $\mathcal{P}$.*

Following the same arguments as in the proof of Theorem 17, we have the following result.

Theorem 18. *For a set $\mathscr{S}$ of n line segments, the compact representation of the nearest-site Voronoi diagram $NVD_{\mathcal{P}}(\mathscr{S})$ can be computed in $O(n(\log n + \log^2 m) + m^2)$ time, where m is the number of sides of the underlying convex polygon $\mathcal{P}$.*

References

1. Abel, Z., Demaine, E.D., Demaine, M.L., Itoh, J.-I., Lubiw, A., Nara, C., O'Rourke, J.: Continuously flattening polyhedra using straight skeletons. In: 13th Symposium on Computational Geometry (SoCG), pp. 396:396–396:405 (2014)
2. Aichholzer, O., Aurenhammer, F.: Straight skeletons for general polygonal figures in the plane. In: COCOON, pp. 117–126 (1996)
3. Aurenhammer, F.: Voronoi diagrams - a survey of a fundamental geometric data structure. ACM Comput. Surv. **23**(3), 345–405 (1991)
4. Aurenhammer, F., Drysdale, R.L.S., Krasser, H.: Farthest line segment Voronoi diagrams. Inf. Process. Lett. **100**(6), 220–225 (2006)
5. Aurenhammer, F., Klein, R., Lee, D.: Voronoi Diagrams and Delaunay Triangulations. World Scientific, Singapore (2013)
6. Barequet, G., De, M.: Voronoi diagram for convex polygonal sites with convex polygon-offset distance function. In: Gaur, D., Narayanaswamy, N.S. (eds.) CALDAM 2017. LNCS, vol. 10156, pp. 24–36. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-53007-9_3
7. Barequet, G., Dickerson, M.T., Goodrich, M.T.: Voronoi diagrams for convex polygon-offset distance functions. Discret. Comput. Geom. **25**(2), 271–291 (2001)
8. Barequet, G., Goodrich, M.T., Levi-Steiner, A., Steiner, D.: Contour interpolation by straight skeletons. Graph. Models **66**(4), 245–260 (2004)
9. Bohler, C., Cheilaris, P., Klein, R., Liu, C., Papadopoulou, E., Zavershynskyi, M.: On the complexity of higher order abstract Voronoi diagrams. Comput. Geom. **48**(8), 539–551 (2015)
10. Cheng, S.-W., Mencel, L., Vigneron, A.: A faster algorithm for computing straight skeletons. In: Schulz, A.S., Wagner, D. (eds.) ESA 2014. LNCS, vol. 8737, pp. 272–283. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44777-2_23
11. Cheong, O., Everett, H., Glisse, M., Gudmundsson, J., Hornus, S., Lazard, S., Lee, M., Na, H.: Farthest-polygon Voronoi diagrams. Comput. Geom. **44**(4), 234–247 (2011)
12. Fortune, S.: A sweepline algorithm for Voronoi diagrams. Algorithmica **2**, 153–174 (1987)
13. Huber, S., Held, M.: A fast straight-skeleton algorithm based on generalized motorcycle graphs. Int. J. Comput. Geom. Appl. **22**(05), 471–498 (2012)
14. Kirkpatrick, D.G.: Efficient computation of continuous skeletons. In: 20th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 18–27 (1979)
15. Klein, R., Wood, D.: Voronoi diagrams based on general metrics in the plane. In: Cori, R., Wirsing, M. (eds.) STACS 1988. LNCS, vol. 294, pp. 281–291. Springer, Heidelberg (1988). https://doi.org/10.1007/BFb0035852
16. Leven, D., Sharir, M.: Planning a purely translational motion for a convex object in two-dimensional space using generalized Voronoi diagrams. Discret. Comput. Geom. **2**, 9–31 (1987)
17. McAllister, M., Kirkpatrick, D.G., Snoeyink, J.: A compact piecewise-linear Voronoi diagram for convex sites in the plane. Discret. Comput. Geom. **15**(1), 73–105 (1996)
18. Papadopoulou, E., Dey, S.K.: On the farthest line-segment Voronoi diagram. Int. J. Comput. Geom. Appl. **23**(6), 443–460 (2013)
19. Papadopoulou, E., Zavershynskyi, M.: The higher-order Voronoi diagram of line segments. Algorithmica **74**(1), 415–439 (2016)
20. Yap, C.: An $O(n \log n)$ algorithm for the Voronoi diagram of a set of simple curve segments. Discret. Comput. Geom. **2**, 365–393 (1987)