Approximate Euclidean Shortest Paths in Polygonal Domains

R. Inkulu
Department of Computer Science & Engineering, IIT Guwahati, India
http://www.iitg.ac.in/rinkulu/
rinkulu@iitg.ac.in

Sanjiv Kapoor
Department of Computer Science & Engineering, IIT Chicago, USA
http://www.cs.iit.edu/~kapoor/
kapoor@iit.edu

─── **Abstract** ───

Given a set $\mathcal{P}$ of h pairwise disjoint simple polygonal obstacles in $\mathbb{R}^2$ defined with n vertices, we compute a sketch Ω of $\mathcal{P}$ whose size is independent of n, depending only on h and the input parameter ϵ. We utilize Ω to compute a $(1+\epsilon)$-approximate geodesic shortest path between the two given points in $O(n + h((\lg n) + (\lg h)^{1+\delta} + (\frac{1}{\epsilon} \lg \frac{h}{\epsilon})))$ time. Here, ϵ is a user parameter, and δ is a small positive constant (resulting from the time for triangulating the free space of $\mathcal{P}$ using the algorithm in [3]). Moreover, we devise a $(2+\epsilon)$-approximation algorithm to answer two-point Euclidean distance queries for the case of convex polygonal obstacles.

1 Introduction

For any set $\mathcal{Q}$ of pairwise-disjoint simple polygonal obstacles in $\mathbb{R}^2$, the free space $\mathcal{F}(\mathcal{Q})$ is the closure of $\mathbb{R}^2$ without the union of the interior of all the polygons in $\mathcal{Q}$. Given a set $\mathcal{P} = \{P_1, P_2, \ldots, P_h\}$ of pairwise-disjoint simple polygonal obstacles in $\mathbb{R}^2$ and two points s and t in $\mathcal{F}(\mathcal{P})$, the *Euclidean shortest path finding problem* seeks to compute a shortest path between s and t that lies in $\mathcal{F}(\mathcal{P})$. This problem is well-known in the computational geometry community. Mitchell [10, 27] provides an extensive survey of research accomplished in determining shortest paths in polygonal and polyhedral domains. The problem of finding shortest paths in graphs is quite popular and considered to be fundamental. Especially, several algorithms for efficiently computing single-source shortest paths and all-pairs shortest paths are presented in Cormen et al. [9] and Kleinberg and Tardos [25]) texts. And, the algorithms for approximate shortest paths are surveyed in [28]. In the following, we assume that n vertices together define the h polygonal obstacles of $\mathcal{P}$.

Given a polygonal domain $\mathcal{P}$ as input, the following are three well-known variants of the Euclidean shortest path finding problem: (i) both s and t are given as input with $\mathcal{P}$, (ii) only s is provided as input with $\mathcal{P}$, and (iii) neither s nor t is given as input. The type (i) problem is a single-shot problem and involves no preprocessing. The preprocessing phase of the algorithm for a type (ii) problem constructs a shortest path map with s as the source so that a shortest path between s and any given query point t can be found efficiently. In the

third variation, which is known as a two-point shortest path query problem, the polygonal domain $\mathcal{P}$ is preprocessed to construct data structures that facilitate in answering shortest path queries between any given pair of query points s and t.

In solving a type (i) or type (ii) problem, there are two fundamentally different approaches: the visibility graph method (see Ghosh [14] for both the survey and details of various visibility algorithms) and the continuous Dijkstra (wavefront propagation) method. The visibility graph method [5, 21, 22, 29] is based on constructing a graph G, termed visibility graph, whose nodes are the vertices of the obstacles (together with s and t) and edges are the pairs of mutually visible vertices. Once the visibility graph G is available, a shortest path between s and t in G is found using Dijkstra's algorithm. As the number of edges in the visibility graph is $O(n^2)$, this method has worst-case quadratic time complexity. In the continuous Dijkstra approach [17, 18, 19, 26], a wavefront is expanded from s till it reaches t. In specific, for the case of polygonal obstacles in plane, Hershberger and Suri devised an algorithm in [17] which computes a shortest path in $O(n \lg n)$ time and the algorithm in [18] (which extends the algorithm by Kapoor [19]) by Inkulu, Kapoor, and Maheshwari computes a shortest path in $O(n + h((\lg h)^\delta + (\lg n)(\lg h)))$ time. Here, δ is a small positive constant (resulting from the time for triangulating the $\mathcal{F}(\mathcal{P})$ using the algorithm in [3]). The continuous Dijkstra method typically constructs a shortest path map with respect to s so that for any query point t, a shortest path from s to t can be found efficiently.

The two-point shortest path query problem within a given simple polygon was addressed by Guibas and Hershberger [15]. It preprocessed the simple polygon in $O(n)$ time and constructed a data structure of size $O(n)$ and answers two-point shortest distance queries in $O(\lg n)$ time. Exact two-point shortest path queries in the polygonal domain were explored by Chiang and Mitchell [7]. One of the algorithms in [7] constructs data structures of size $O(n^5)$ and answers the query for any two-point distance in $O(h + \lg n)$ worst-case time. And, another algorithm in [7] builds data structures of size $O(n + h^5)$ and outputs any two-point distance query in $O(h \lg n)$ time. In both of these algorithms, a shortest path itself is found in additional time $O(k)$, where k is the number of edges in the output path. Guo et al. [16] preprocessed $\mathcal{F}(\mathcal{P})$ in $O(n^2 \lg n)$ time to compute data structures of size $O(n^2)$ for answering two-point distance queries for any given pair of query points in $O(h \lg n)$ time.

Because of the difficulty of exact two-point queries in polygonal domains, various approximation algorithms were devised. Clarkson first made such an attempt in [8]. Chen [4] used the techniques from [8] in constructing data structures of size $O(n \lg n + \frac{n}{\epsilon})$ in $o(n^{3/2}) + O(\frac{n}{\epsilon} \lg n)$ time to support $(6 + \epsilon)$-approximate two-point distance queries in $O(\frac{1}{\epsilon} \lg n + \frac{1}{\epsilon^2})$ time, and a shortest path in additional $O(L)$ time, where L is the number of edges of the output path. Arikati et al. [2] devised a family of algorithms to answer two-point approximate shortest path queries. Their first algorithm outputs a $(\sqrt{2} + \epsilon)$-approximate distance; depending on a parameter $1 \le r \le n$, in the worst-case, either the preprocessed data structures of this algorithm take $O(n^2)$ space or the query time is $O(\sqrt{n})$. Their second algorithm takes $O(n)$ query time to report the distance. The stretch of the third and fourth algorithms proposed in [2] are respectively $(2\sqrt{2} + \epsilon)$ and $(3\sqrt{2} + \epsilon)$. Agarwal et al. [1] computes a $(1 + \epsilon)$-approximate geodesic shortest path in $O(n + \frac{h}{\sqrt{\epsilon}} \lg(\frac{h}{\epsilon}))$ time when the obstacles are convex.

Throughout this paper, to distinguish graph vertices from the vertices of the polygonal domain, we refer to vertices of a graph as nodes. The Euclidean distance between any two points p and q is denoted with $\|pq\|$. The obstacle-avoiding geodesic Euclidean shortest path distance between any two points p, q amid a set $\mathcal{Q}$ of obstacles is denoted with $dist_{\mathcal{Q}}(p, q)$. The (shortest) distance between two nodes s and t in a graph G is denoted with $dist_G(s, t)$. Unless specified otherwise, distance is measured in Euclidean metric. We denote both the

convex hull of a set R of points and the convex hull of a simple polygon R with $CH(R)$. Let r' and r'' be two rays with origin at p. Let $\vec{v_1}$ and $\vec{v_2}$ be the unit vectors along the rays r' and r'' respectively. A *cone* $C_p(r', r'')$ is the set of points defined by rays r' and r'' such that a point $q \in C_p(r', r'')$ if and only if q can be expressed as a convex combination of the vectors $\vec{v_1}$ and $\vec{v_2}$ with positive coefficients. When the rays are evident from the context, we denote the cone with C_p. The counterclockwise angle from the positive x-axis to the line that bisects the cone angle of C_p is termed as the *orientation of the cone C_p*.

Our contributions

First, we describe the algorithm for the case in which $\mathcal{P}$ comprises convex polygonal obstacles. We compute a sketch Ω from the polygonal domain $\mathcal{P}$. Essentially, each convex polygonal obstacle P_i in $\mathcal{P}$ is approximated with another convex polygonal obstacle whose complexity depends only on the input parameter ϵ; significantly, the size of the approximated polygon is independent of the size of P_i. In specific, when $\mathcal{P}$ is comprised of h convex polygonal obstacles, the sketch Ω is comprised of h convex polygonal obstacles: for each $1 \leq i \leq h$, the convex polygon $P_i \in \mathcal{P}$ is approximated with another convex polygon $Q_i \in \Omega$. For each $P_i \in \mathcal{P}$, we identify a coreset S_i of vertices of P_i and form the core-polygon $Q_i \in \Omega$ using S_i. When P_i is convex, the corresponding core-polygon Q_i obtained through this procedure is convex; and, $Q_i \subseteq P_i$. Like in [1], the combinatorial complexity of Ω is independent of n; it depends only on h and the input parameter ϵ. For two points $s, t \in \mathcal{F}(\mathcal{P})$, we compute an approximate Euclidean shortest path between s and t in $\mathcal{F}(\Omega)$ using an algorithm that is a variant of [8]. From this path, we compute a path R in $\mathcal{F}(\mathcal{P})$ and show that R is a $(1 + \epsilon)$-approximate Euclidean shortest path between s and t amid polygonal obstacles in $\mathcal{P}$. When the obstacles in $\mathcal{P}$ are not necessarily convex, we compute the sketch of $\mathcal{P}$ using the convex chains (that bound the obstacles) as well as the corridor paths that result from the hourglass decomposition [19, 20, 22] of $\mathcal{F}(\mathcal{P})$. The main contributions and the major advantages in our approach are described in the following:

- When $\mathcal{P}$ is comprised of disjoint simple polygonal obstacles, we compute a $(1 + \epsilon)$-approximate geodesic Euclidean shortest path between the two given points belonging to $\mathcal{F}(\mathcal{P})$ in $O(n + h((\lg n) + (\lg h)^{1+\delta} + \frac{1}{\epsilon} \lg \frac{h}{\epsilon}))$ time. Here, δ is a small positive constant resulting from the triangulation of the free space using the algorithm from [3]. (Refer to Theorem 9.) Agarwal et al. [1] compute a $(1 + \epsilon)$-approximate geodesic shortest path in $O(n + \frac{h}{\sqrt{\epsilon}} \lg(\frac{h}{\epsilon}))$ time when the obstacles are convex. In computing approximate shortest paths, our algorithm extends the notion of coresets in [1] to simple polygons. However, our approach is computing coresets, and an approximate shortest path using these coresets is quite different from [1]. Our algorithm to construct the sketch of $\mathcal{P}$ is simpler.

- As part of devising the above algorithm, when $\mathcal{P}$ is comprised of convex polygonal obstacles, our algorithm computes a $(1 + \epsilon)$-approximate geodesic Euclidean distance between the two given points in $O(n + \frac{h}{\epsilon} \lg \frac{h}{\epsilon})$ time. Further, our algorithm computes a $(1 + \epsilon)$-approximate shortest path in additional $O(h \lg n)$ time. (Refer to Theorem 7.)

- When $\mathcal{P}$ is comprised of disjoint convex polygonal obstacles, we preprocess these polygons in $O(n + \frac{h}{\epsilon^2}(\lg \frac{h}{\epsilon}) + \frac{h}{\epsilon}(\lg \frac{h}{\epsilon})^2)$ time to construct data structures of size $O(\frac{h}{\epsilon})$ for answering any two-point $(2 + \epsilon)$-approximate geodesic distance (length) query in $O(\frac{1}{\epsilon^6}(\lg \frac{h}{\epsilon})^2)$ time. (Refer to Theorem 12.) To compute an optimal geodesic shortest path amid simple polygonal obstacles, Chen and Wang [5] takes $O(n + h \lg h + k)$ time, where k is a parameter sensitive to the geometric structures of the input and is upper bounded by $O(h^2)$. Our algorithm to answer approximate two-point distance queries amid convex polygonal obstacles takes space close to linear in n whereas the preprocessed data

structures of algorithms proposed in [7] occupy $\Omega(n^5)$ space in the worst-case. Also, our algorithm for two-point distance queries improves the stretch factor of [4] from $(6 + \epsilon)$ to $(2 + \epsilon)$ in case of convex polygonal obstacles.

- Furthermore, our algorithm to compute the coreset of simple polygons to obtain a sketch Ω of $\mathcal{P}$ as well as the algorithm to compute an approximate geodesic Euclidean shortest path in $\mathcal{P}$ using the sketch Ω may be of independent interest.

Section 2 describes an algorithm for computing a single-shot approximate shortest path when obstacles in $\mathcal{P}$ are convex polygons. Section 3 extends this algorithm to compute an approximate Euclidean shortest path amid simple polygonal obstacles. The algorithm to answer two-point approximate Euclidean distance queries amid convex polygonal obstacles is described in Section 4. A table comparing earlier algorithms to ours is given in the Appendix.

## 2 Approximate shortest path amid convex polygons

In this section, we consider the case in which every simple polygon in $\mathcal{P}$ is convex. We use the following notation from Yao [30]. Let $\kappa \geq 2$, and define $\theta = 2\pi/\kappa$. Consider the set of κ rays: for $0 \leq i\kappa$, the ray r_i passes through the origin and makes an angle $i\theta$ with the positive x-axis. Each pair of successive rays defines a cone whose apex is at the origin. This collection of κ cones is denoted by $\mathcal{C}$. It is clear that the cones of $\mathcal{C}$ partition the plane. Also, the two bounding rays of any cone of $\mathcal{C}$ make an angle θ. In our algorithm, the value of κ is chosen as a function of ϵ (refer to Subsection 2.2). When a cone $C \in \mathcal{C}$ is translated to have the apex at a point p, the translated cone is denoted with C_p. Each cone that we refer in this paper is a translated copy of some cone in $\mathcal{C}$. For each polygon P in $\mathcal{P}$, we choose a subset of $O(\frac{1}{\sqrt{\alpha\epsilon}})$ vertices from the vertices of P. At each such vertex p, we introduce a set of cones at p.In the algorithm to compute a single-shot s-t geodesic shortest path, the value of α is set to $\frac{\epsilon}{2}$. The algorithm for two-point approximate distance queries sets the value of α to $\frac{\epsilon}{12}$. The proof of Theorem 7 details the reasons for setting these specific values. As detailed below, these vertices and cones help in computing a spanner that approximates a Euclidean shortest path between the two given points in $\mathcal{F}(\mathcal{P})$.

2.1 Sketch of $\mathcal{P}$

In this subsection, we define and characterise the sketch of $\mathcal{P}$. For any $P_i \in \mathcal{P}$ and any two points p' and p'' on the boundary of P_i, the section of boundary of P_i that occurs while traversing from p' to p'' in counterclockwise order is termed a *patch* of P_i. In specific, we partition the boundary of each $P_i \in \mathcal{P}$ into a collection of patches Γ_i such that for any two points p', p'' belonging to any patch $\gamma \in \Gamma_i$, the angle between the outward (w.r.t. the centre of P_i) normals to respective edges at p' and p'' is upper bounded by $\frac{\sqrt{\alpha\epsilon}}{2}$. The maximum angle between the outward normals to any two edges that belong to a patch γ constructed in our algorithm is the *angle subtended by γ*. To facilitate in computing patches of any obstacle P_i, we partition the unit circle $\mathbb{S}^2$ centred at the origin into a minimum number of segments such that each circular segment is of length at most $\frac{\sqrt{\alpha\epsilon}}{2}$. For every such segment s of $\mathbb{S}^2$, a patch (corresponding to s) comprises of the maximal set of the contiguous sequence of edges of P_i whose outward normals intersect s, when each of these normals is translated to the origin. (To avoid degeneracies, we assume each normal intersects a single segment.) Let Γ_i be a partition of the boundary of a convex polygon P_i into a collection of $O(\frac{1}{\sqrt{\alpha\epsilon}})$ patches. The lemma below shows that the geodesic distance between any two points belonging to any patch $\gamma \in \Gamma_i$ is a $(1 + \alpha\epsilon)$-approximation to the Euclidean distance between them.

▶ **Lemma 1.** *For any two points p and q that belong to any patch $\gamma \in \Gamma_j$, the geodesic distance between p and q along γ is upper bounded by $(1 + \alpha\epsilon)\|pq\|$ for $\alpha\epsilon < 1$.*

Proof. Let e' be the edge on which p lies and let e'' be the edge on which q lies. Let c be the point of intersection of normal to e' at p and the normal to e'' at q. Since p and
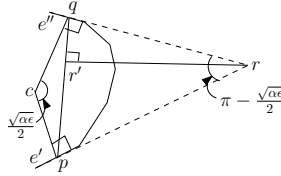
■ **Figure 1** Illustrating the construction in proving the upper bound on the patch length.

q belong to the same patch, the angle between cp and cq is upper bounded by $\frac{\sqrt{\alpha\epsilon}}{2}$, when the value of $\alpha\epsilon$ is small. Let l' and l'' be the lines that respectively pass through e' and e''. Also, let r be the point at which lines l' and l'' intersect. (Refer to Fig. 1.) For the small values of $\sqrt{\alpha\epsilon}$ and due to triangle inequality, the geodesic length of the patch between p and q is upper bounded by $\|pr\| + \|qr\|$. Let r' be the point of projection of r on to line segment pq. Suppose $\angle qrr' = \angle prr'$. (Analysis of other cases is similar.) Then $\|pr\| + \|qr\| \leq \frac{\|pr'\|}{\sin(\frac{\pi}{2} - \frac{\sqrt{\alpha\epsilon}}{4})} + \frac{\|r'q\|}{\sin(\frac{\pi}{2} - \frac{\sqrt{\alpha\epsilon}}{4})} = \frac{\|pq\|}{\cos\frac{\sqrt{\alpha\epsilon}}{4}} \leq (1 + \alpha\epsilon)\|pq\|$. The last inequality is valid when $\alpha\epsilon < 1$. ◀

For each obstacle P_i, the *coreset S_i of P_i* is comprised of two vertices chosen from each patch in Γ_i. In particular, for each patch $\gamma \in \Gamma_i$, the first and last vertices of γ that occur while traversing the boundary of P_i are chosen to be in the coreset S_i of P_i. The *coreset $\mathcal{S}$ of $\mathcal{P}$* is then simply $\bigcup_i S_i$.

▶ **Observation 1.** *The size of the coreset $\mathcal{S}$ of $\mathcal{P}$ is $O(\frac{h}{\sqrt{\alpha\epsilon}})$.*

For every $1 \leq i \leq h$, our algorithm uses *core-polygon $Q_i = CH(S_i)$* in place of P_i. For any single point obstacle P_i in $\mathcal{P}$, the core-polygon of P_i is that point itself. The patch construction procedure guarantees that each polygonal obstacle in $\mathcal{P}$ is partitioned into patches such that the core-polygon that correspond to every obstacle in $\mathcal{P}$ is valid. Let Ω be the set comprising of core-polygons corresponding to each of the polygons in $\mathcal{P}$. The set Ω is called the *sketch of $\mathcal{P}$*. The following lemmas show that Ω facilitates in computing a $(1 + \alpha\epsilon)$-approximation of the geodesic distance between any two given points in $\mathcal{F}(\mathcal{P})$.

▶ **Lemma 2.** *Let v', v'' be any two vertices of obstacles in Ω. Then, $dist_{\mathcal{P}}(v', v'')$ is upper bounded by $(1 + \alpha\epsilon)dist_{\Omega}(v', v'')$.*

Proof. Let v_1, v_2 be any two successive vertices along a shortest path between v' and v'' in $\mathcal{F}(\Omega)$. Let $\mathcal{O} \subseteq \mathcal{P}$ be the set of obstacles intersected by the line segment v_1v_2. Let v_1 and v_2 be respectively belonging to obstacles P_j and P_k. Also, let Γ_j (resp. Γ_k) be the set comprising the partition of boundary of P_j (resp. P_k) into patches. And, let S_j (resp. S_k) be the coreset of P_j (resp. P_k). Since the line segment v_1v_2 does not intersect the interior of the $CH(S_j)$ or $CH(S_k)$, it intersects at most one patch belonging to set Γ_j and at most one patch belonging to set Γ_k. Let v_1 and r be the points of intersection of line segment v_1v_2 with a patch $\gamma \in \Gamma_j$. (These points might as well be the endpoints of γ.) Then from Lemma 1, the geodesic distance between v_1 and r along γ is upper bounded by $(1 + \alpha\epsilon)\|v_1r\|$. (Refer Fig. 2.) Analogously, let v_2 and r' be the points of intersection of line segment v_1v_2

with a patch $\gamma' \in \Gamma_k$. Then the geodesic distance between v_2 and r' is upper bounded by $(1 + \alpha\epsilon)\|v_2 r'\|$. For any convex polygonal obstacle P_l in $\mathcal{O}$ distinct from P_j and P_k, let p', p'' be the points of intersection of $v_1 v_2$ with the boundary of P_l. Since the line segment $v_1 v_2$ does not intersect the interior of the convex hull of coreset corresponding to P_l, both p' and p'' belong to the same patch, say $\gamma'' \in \Gamma_l$. Then again from Lemma 1, the geodesic distance between p' and p'' along patch γ'' is upper bounded by $(1 + \alpha\epsilon)\|p'p''\|$. We modify $v_1 v_2$ as follows: For every maximal subsection, say $p'_i p''_i$, of the line segment $v_1 v_2$ that is interior to a polygonal obstacle of $\mathcal{P}$, we replace that subsection with a geodesic Euclidean shortest path in $\mathcal{F}(\mathcal{P})$ between p'_i and p''_i.
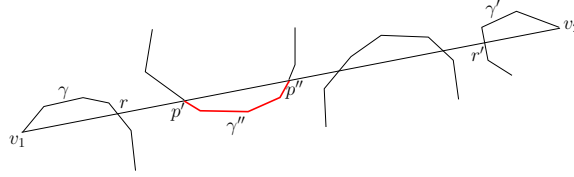
■ **Figure 2** A line segment $v_1 v_2$ of a shortest path amid Ω intersecting three patches belonging to obstacles in $\mathcal{P}$.

Let $\gamma_1, \gamma_2, \ldots, \gamma_k$ be the set Γ of patches intersected by the line segment $v_1 v_2$. Also, for every $1 \leq i \leq k$, let p'_i, p''_i be the points of intersections of $v_1 v_2$ with patch $\gamma_i \in \Gamma$ with p'_i closer to v_1 than v_2 along the line segment $v_1 v_2$. Then $\sum_{i=1}^{k} dist_{\mathcal{P}}(p'_i, p''_i)$ added with $\sum_{i=1}^{k-1} \|p''_i p'_{i+1}\|$ is upper bounded by $(1 + \alpha\epsilon)\|v_1 v_2\|$. Let $v_1, \ldots, v_l$ be the vertices of $\mathcal{P}$ that occur in that order along a Euclidean shortest path in $\mathcal{F}(\Omega)$ between vertices $v', v'' \in \mathcal{P}$. Then $dist_{\mathcal{P}}(v_1, v_l) = \sum_{i=1}^{l-1} dist_{\mathcal{P}}(v_i, v_{i+1}) \leq (1 + \alpha\epsilon) \sum_{i=1}^{l-1} dist_{\Omega}(v_i, v_{i+1})$. Note that we do this transformation for each line segment of the shortest path that intersects any patch. ◄

Since $\mathcal{F}(\mathcal{P}) \subseteq \mathcal{F}(\Omega)$, every path that avoids convex polygonal obstacles in $\mathcal{P}$ is also a path that avoids convex polygonal obstacles in Ω. This observation leads to the following:

▶ **Lemma 3.** *For any two vertices v', v'' of $\mathcal{P}$, $dist_{\Omega}(v', v'') \leq dist_{\mathcal{P}}(v', v'')$.*

Considering the given two points $s, t \in \mathcal{F}(\mathcal{P})$ as degenerate obstacles, a $(1 + \alpha\epsilon)$-approximation of the shortest distance between s and t amid polygonal obstacles in $\mathcal{P}$ is computed.

▶ **Lemma 4.** *For a set $\mathcal{P}$ of h pairwise disjoint convex polygons in $\mathbb{R}^2$ and two points $s, t \in \mathcal{F}(\mathcal{P})$, the sketch $\mathcal{S}$ of $\mathcal{P}$ with cardinality $O(\frac{h}{\sqrt{\alpha\epsilon}})$ suffices to compute a $(1 + \alpha\epsilon)$-approximate shortest path between s and t in $\mathcal{F}(\mathcal{P})$.*

Proof. Immediate from Observation 1, Lemma 2, and Lemma 3. ◄

Our approach in computing coresets and an approximate shortest path using these coresets is quite different from [1]. As will be shown in Section 3, our sketch construction is extended to compute shortest paths even when $\mathcal{P}$ comprises of polygon obstacles which are not necessarily convex. How our algorithm differs from [1] for the convex polygonal case is detailed herewith. Let $\mathcal{P}$ be the polygonal domain defined with convex polygons $P_1, P_2, \ldots, P_h$. In this algorithm as well as in [1], P_i is approximated with Q_i, for every $1 \leq i \leq h$. However, for every $1 \leq i \leq h$, in our algorithm $Q_i \subseteq P_i$ whereas in [1], $P_i \subseteq Q_i$. Let the new polygonal domain Ω be defined with simple polygons $Q_1, Q_2, \ldots, Q_h$. Unlike [1], in computing Ω, our algorithm does not require using plane sweep algorithm to find pairwise vertically visible simple polygons of $\mathcal{P}$. As described above, our algorithm partitions the boundary of each convex polygon P into a set of patches.
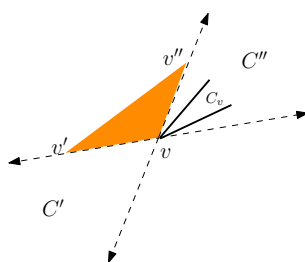
■ **Figure 3** Illustrating an admissible cone C_v incident to a coreset vertex v of an obstacle.

2.2 Computing an approximate geodesic shortest path in $\mathcal{F}(\mathcal{P})$ using the sketch Ω of $\mathcal{P}$

Since we intend to compute an approximate shortest path, to keep our algorithm simpler, we do not want to use the algorithm from [17] to compute a shortest path amid convex polygonal obstacles in Ω, Instead, we use a spanner constructed with the conic Voronoi diagrams (CVDs) [8]. Further, in our algorithm, for any maximal line segment with endpoints r', r'' along the computed (approximate) shortest path amid obstacles in Ω, if the line segment $r'r''$ lies in $\mathcal{F}(\mathcal{Q}) - \mathcal{F}(\mathcal{P})$, we replace line segment $r'r''$ with the geodesic Euclidean shortest path between r' and r'' in $\mathcal{F}(\mathcal{P})$.

Since our algorithm relies on [8], we give a brief overview of that algorithm first. The algorithm in [8] constructs a spanner $G(V, E)$ for polygonal domain $\mathcal{P}$. Noting that the endpoints of line segments of a shortest path in $\mathcal{F}(\mathcal{P})$ are a subset of vertices of polygonal obstacles in $\mathcal{P}$, the node set V is defined as the vertex set of $\mathcal{P}$. Let $\mathcal{C}'$ be the set of $O(\frac{1}{\epsilon})$ cones with apex at the origin of the coordinate system together partitioning $\mathbb{R}^2$. (The cone angle of each cone in $\mathcal{C}'$ except for one is set to ϵ and that one cone has $2\pi - \lfloor \frac{2\pi}{\epsilon} \rfloor \epsilon$ as the cone angle.) Let $C \in \mathcal{C}'$ be a cone with orientation θ and let $C' \in \mathcal{C}'$ be the cone with orientation $-\theta$. For each cone $C \in \mathcal{C}'$ and a set K of points, the set of cones resultant from introducing a cone C_p for every point $p \in K$, is the conic Voronoi diagram $CVD(C, K)$. (Note that as mentioned earlier, C_p is the cone resulted from translating cone C to have the apex at the point p.) For a given cone C_v, among all the points on the boundaries of polygons in $\mathcal{P}$ that are visible from v, a point p whose projection onto the bisector of C_v is closest to v is said to be *a closest point in C_v to v*. If more than one point is closest in C_v to v, then we arbitrarily pick one of those points. For every vertex v of $\mathcal{P}$ and for every cone C_v, if a closest point p in $C_v - \{v\}$ to v is not a vertex of $\mathcal{P}$, then the algorithm includes p as a node in V. Further, for every vertex v of $\mathcal{P}$ and for every cone C_v, an edge e joining v and a closest point p in $C_v - \{v\}$ to v is introduced in E with its weight equal to the Euclidean distance between v and p. For every node v in G that corresponds to a point p on the boundary of $P \in \mathcal{P}$, if p is not a vertex of $\mathcal{P}$, then for every neighbor p' of p on the boundary of P which has a corresponding node v' in V, we introduce an edge e' between v and p' into E and set the weight of e' equal to the Euclidean distance between v and p'. These are the only edges included in E. The Theorem 2.5 in [8] proves that if d is the obstacle-avoiding geodesic Euclidean shortest path distance between any two vertices, say v' and v'', of $\mathcal{P}$, then the distance between the corresponding nodes v' and v'' in G is upper bounded by $(1 + \epsilon)d$. The $CVD(C, K)$ is computed using the plane sweep in $O(|K| \lg |K|)$ time; and, the well-known planar point location data structure is used to locate the region in $CVD(C, K)$ to which a given query point belongs to.

As detailed below, apart from computing a sketch Ω of $\mathcal{P}$, as compared with [8], the number of cones per obstacle that participate in computing CVDs amid $\mathcal{F}(\Omega)$ is further optimized by exploiting the convexity of obstacles together with the properties of shortest paths amid convex obstacles. By limiting the number of vertices of $\mathcal{P}$ at which the cones are initiated to coreset $\mathcal{S}$ of vertices, our algorithm improves the space complexity of the algorithm in [8]. Further, by exploiting the convexity of obstacles, we introduce $O(\frac{1}{\sqrt{\alpha\epsilon}})$ cones per obstacle, each with cone angle $O(\sqrt{\alpha\epsilon})$, and show that these are sufficient to achieve the claimed approximation factor.

Let v be a vertex of $\mathcal{P}$ that belongs to coreset S_i of convex polygon P_i. Let v', v, v'' be the vertices that respectively occur while traversing the boundary of P_i in counterclockwise order. Also, let C' be the cone defined by the pair of rays $(\overrightarrow{vv'}, -\overrightarrow{vv''})$ and let C'' be the cone defined by the pair of rays $(\overrightarrow{vv''}, -\overrightarrow{vv'})$. For a coreset vertex $v \in \mathcal{S}$, a cone $C \in \mathcal{C}$ is said to be *admissible* at v whenever $C_v \cap C'$ or $C_v \cap C''$ is non-empty. (See Fig. 3.) Let p and q be two points in $\mathcal{F}(\mathcal{P})$ such that p and q are not visible to each other due to polygonal obstacles in $\mathcal{P}$. Let v be a vertex of P_i through which a shortest path between p and q passes. Since any shortest path is convex at v with respect to P_i, there exists a shortest path between p and q where one of its line segment lies in C', and another line segment of that path lies in C''. Hence, in computing a Euclidean shortest path amid $\mathcal{P}$, it suffices to consider admissible cones at the vertices of $\mathcal{P}$.

Note that whenever two points s and t between which we intend to find a shortest path are visible to each other, the line segment st needs to be computed. To facilitate this, for every degenerate point obstacle p, every cone C with apex p is considered to be an admissible cone.

The same properties carry over to the polygonal domain Ω as well. For any two points p_1 and p_2 in $\mathcal{F}(\Omega)$, suppose that p_1 and p_2 are not visible to each other. Consider any shortest path τ between p_1 and p_2. For any line segment ab in τ, ab is either an edge of a polygon in Ω or it is a tangent to an obstacle $O \in \Omega$. In the latter case, ab belongs to an admissible cone of O. When the polygonal domain is Ω, the following Lemma upper bounds the number of cones at the vertices of convex polygons in Ω.

▶ **Lemma 5.** *The number of cones introduced at all the obstacles of Ω is $O(\frac{h}{\sqrt{\alpha\epsilon}})$.*

Proof. Let O be the origin of the coordinate system. Let $\overrightarrow{r}$ be a ray with origin at O. (See Fig. 4.) For any two distinct vertices v' and v'' of a convex polygon P, let $\overrightarrow{r_{v'}}$ be the ray parallel to $\overrightarrow{r}$ with origin at v' and pointing in the same direction as $\overrightarrow{r}$ and let $\overrightarrow{r_{v''}}$ be the ray parallel to $\overrightarrow{r}$ with origin at v'' and point in the same direction as $\overrightarrow{r}$. Also, let v_1' precede v' (resp. v_1'' precede v'') and v_2' succeed v' (resp. v_2'' succeed v'') while traversing the boundary of P in counterclockwise order. Since P is a convex polygon, if every point of $\overrightarrow{r_{v'}}$ belongs to the cone defined by $\overrightarrow{v_1'v'}$ and $\overrightarrow{v'v_2'}$ then it is guaranteed that not every point of $\overrightarrow{r_{v''}}$ belongs to the cone defined by $\overrightarrow{v''v_2''}$ and $\overrightarrow{v_1''v''}$. Extending this argument, if a cone $C_{v'}$ is admissible
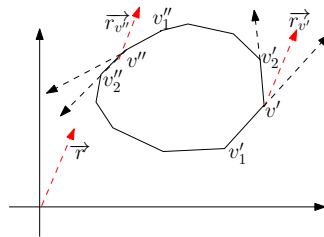
Figure 4 Illustrating that a ray parallel to r can exist in only one admissible cone per obstacle.
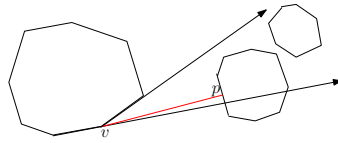
Figure 5 Illustrating an edge of the spanner.

at v' then the cone $C_{v''}$ cannot be admissible at v''. Since the number of coreset vertices per obstacle is $O(\frac{1}{\sqrt{\alpha\epsilon}})$, the number of cones introduced per obstacle is $O(\frac{1}{\sqrt{\alpha\epsilon}})$. Further, since there are h convex polygonal obstacles, number of cones at all the obstacle vertices together is $O(\frac{h}{\sqrt{\alpha\epsilon}})$.   ◀

Next, we describe the algorithm to compute the spanner $G(V = S \cup S', E)$. The set S comprises of nodes corresponding to coreset $\mathcal{S}$. The set S' is a set of Steiner points, as follows. For every $v \in S$ and every admissible cone C_v, let V' be the set of points on the boundaries of obstacles of Ω that are visible from v and belong to cone C_v. (See Fig. 5.) The point p in V' that is closest to v, termed the *closest Steiner point in C_v to v*, is determined and p is added to S'. An edge e between v and p is introduced in E while the Euclidean distance between v and p is set as the weight of e in G. Let p be located on a convex polygonal obstacle P. Further, for every Steiner point p, let v' (resp. v'') be the coreset vertex or Steiner point that lies on the boundary of P and occurs before (resp. after) p while traversing the boundary of P in counterclockwise order. Then an edge e' (resp. e'') between p and v' (resp. p and v'') is introduced in E while the geodesic distance between p and v' (resp. p and v'') along the boundary of P is set as the weight of e' (resp. e'') in G. Note that both $|V|$ and $|E|$ are $O(\frac{h}{\sqrt{\alpha\epsilon}})$. For any two points $s, t \in \mathcal{F}(\Omega)$, the following Lemma upper bounds the $dist_G(s, t)$ in terms of $dist_\Omega(s, t)$.

▶ **Lemma 6.** *Let G be the spanner constructed from Ω. Let $dist_G(p', p'')$ be the distance between p' and p'' in G. Then for any two points $s, t \in \mathcal{F}(\Omega)$, $dist_\Omega(s, t) \leq dist_G(s, t) \leq (1 + \sqrt{\alpha\epsilon})dist_\Omega(s, t)$.*

Proof. Theorem 2.5 of [8] concludes that to achieve $(1 + \alpha\epsilon)$-approximation, $\sin\psi - \cos\psi \leq \frac{-1}{1+\alpha\epsilon}$. Expanding *sine* and *cosine* functions for the first few terms yield $-1 + \psi + \frac{\psi^2}{2!} \leq \frac{-1}{1+\alpha\epsilon}$. Solving the quadratic equation in ψ yields $\psi \leq \alpha\epsilon$. Since we are using cones with cone angle $\sqrt{\alpha\epsilon}$ in our algorithm, a $(1 + \sqrt{\alpha\epsilon})$-approximation is achieved.

We claim that introducing a subset of cones (admissible cones) rather than all the cones as used in [8] does not affect the correctness. Let p and q be the vertices of two convex polygons P_i and P_j respectively. Suppose that pq is a line segment belonging to a shortest path R between vertices s and t of the spanner computed in [8]. Further, suppose that p occurs before q when R is traversed from s to t. If the line along pq supports P_i (resp. P_j) at p (resp. q), then the line segment pq belongs to an admissible cone at p (resp. q). Otherwise, there exists a line segment in the admissible cone with apex either at a vertex of P_i or at a vertex of P_j which would yield a shorter path from source s to q without using the line segment pq.   ◀

Once we find a shortest path SP_Ω between s and t amid convex polygonal obstacles in Ω using the spanner G, following the proof of Lemma 3, we transform SP_Ω to a path amid obstacles in $\mathcal{P}$. Since there are $O(h)$ obstacles in Ω, SP_Ω contains $O(h)$ tangents between obstacles. Let this set of tangents be $\mathcal{T}$. We need to find points of intersection of convex polygons in $\mathcal{P}$ with the line segments in $\mathcal{T}$. For any $l \in \mathcal{T}$ and $P_i \in \mathcal{P}$, by using the algorithm from Dobkin et al. [12], we compute the possible intersection between l and P_i. Whenever

a line segment $l \in \mathcal{T}$ and a convex polygon $P_i \in \mathcal{P}$ intersect, say at points p' and p'', we replace the line segment between p' and p'' with the geodesic shortest path between p' and p'' along the boundary of P_i. Analogously, for every line segment $l \in SP_\Omega - \mathcal{T}$ belonging to an obstacle $P_j \in \Omega$, we replace l with the corresponding geodesic path along the boundary of P_j. We use the plane sweep technique [11] to determine whichever line segments in $\mathcal{T}$ could intersect with the convex obstacles in $\mathcal{P}$. Essentially, the event handling procedures of plane sweep algorithm replace every line segment in SP_Ω that intersects with any obstacle $P_j \in \mathcal{P}$ with the shortest geodesic shortest path along the boundary of P_j, so that the resulting shortest path $SP_\mathcal{P}$ after all such replacements belongs to $\mathcal{F}(\mathcal{P})$.

As part of the plane sweep, a vertical line is swept from left-to-right in the plane. Let L (resp. R) be the set of leftmost (resp. rightmost) vertices of convex polygons in $\mathcal{P}$. Initially, points in L and R together with the two endpoints of every line segment in $\mathcal{T}$ are inserted into the priority queue Q. The event points are scheduled from Q using their respective distances from the initial sweep line position. As the events occur, the event points corresponding to L, R, and the endpoints of line segments in $\mathcal{T}$ are handled and are deleted from Q. The algorithm terminates whenever Q is empty. As described below, the intersection points between the line segments in $\mathcal{T}$ and the convex polygons in $\mathcal{P}$ are added to Q with the traversal of the sweep line. The sweep line status is maintained as a balanced binary search tree B. We insert (resp. delete) a pointer to a line segment in $\mathcal{T}$ or a pointer to a convex polygon in $\mathcal{P}$ to B whenever leftmost (resp. rightmost) endpoint of it is popped from Q. We note that before a line segment $l \in \mathcal{T}$ and $P \in \mathcal{P}$ intersect, it is guaranteed that l and P occur adjacent along the sweep line. Hence, whenever l and P are adjacent in the sweep line status, we update the event-point schedule with the point of intersection between l and P that occurs first among all such points of intersection in traversing the sweep line from left to right. By using the algorithm from Dobkin et al. [12], we compute the possible intersection between l and P. If they do intersect, we push the leftmost point of their intersection to Q with the distance from the initial sweep line as the priority of that event point. Further, we store the rightmost intersection point between l and P with the leftmost point of intersection as satellite data. If the leftmost intersection point between l and P pops from Q, we compute the geodesic shortest path along the boundary of P between the leftmost intersection point and the corresponding rightmost intersection point. Further, whenever l and P become non-adjacent along the sweep line, we delete their leftmost point of intersection from Q.

▶ **Theorem 7.** *Given a set $\mathcal{P}$ of pairwise disjoint convex polygons, two points $s, t \in \mathcal{F}(\mathcal{P})$, and $\epsilon \in (0, 0.6]$, computing a $(1 + \epsilon)$-approximate geodesic distance between s and t takes $O(n + \frac{h}{\epsilon} \lg \frac{h}{\epsilon})$ time. Further, within an additional $O(h \lg n)$ time, a $(1 + \epsilon)$-approximate shortest path is computed.*

Proof. From Lemma 4, we know that $dist_\Omega(s,t) \leq dist_\mathcal{P}(s,t) \leq (1 + \alpha\epsilon)dist_\Omega(s,t)$. Let G be the spanner constructed. From Lemma 6, we know that $dist_\Omega(s,t) \leq dist_G(s,t) \leq (1 + \sqrt{\alpha\epsilon})dist_\Omega(s,t)$. As detailed in Lemma 2, algorithm transforms a shortest path between s and t in G to a path p in $\mathcal{F}(\mathcal{P})$. Let $dist_\mathcal{P}^p(s,t)$ be the distance along p. From Lemma 2, $dist_\mathcal{P}^p(s,t) \leq (1 + \alpha\epsilon)dist_G(s,t)$. Hence, $dist_\mathcal{P}^p(s,t) \leq (1 + \alpha\epsilon)dist_G(s,t) \leq (1 + \alpha\epsilon)(1 + \sqrt{\alpha\epsilon})dist_\Omega(s,t) \leq (1 + \alpha\epsilon)(1 + \sqrt{\alpha\epsilon})dist_\mathcal{P}(s,t)$. Since p is a path in $\mathcal{F}(\mathcal{P})$, it is immediate to note that $dist_\mathcal{P}(s,t) \leq dist_\mathcal{P}^p(s,t)$. Therefore, $dist_\mathcal{P}(s,t) \leq dist_\mathcal{P}^p(s,t) \leq (1 + \alpha\epsilon)(1 + \sqrt{\alpha\epsilon})dist_\mathcal{P}(s,t)$. To achieve $(1 + \epsilon)$-approximation, $(1 + \alpha\epsilon)(1 + \sqrt{\alpha\epsilon})$ needs to be less than or equal to $(1 + \epsilon)$. For small values of ϵ ($\epsilon \in (0, 0.6]$), choosing $\alpha = \frac{\epsilon}{2}$ satisfies this inequality.

From here on, we denote $\alpha\epsilon$ with ϵ'. Finding the coreset $\mathcal{S}$ of vertices from the convex polygons in $\mathcal{P}$, and computing the set Ω of core-polygons together takes $O(n)$ time. The number of coreset vertices is $O(\frac{h}{\sqrt{\epsilon'}})$. The number of cones per obstacle is $O(\frac{1}{\sqrt{\epsilon'}})$. Therefore,

the total number of cones is $O(\frac{h}{\sqrt{\epsilon'}})$. For any cone $C \in \mathcal{C}$ and for any core-polygon $O \in \Omega$, at most a constant number of vertices of O are apexes to cones that have the orientation of C. Considering a sweep line in the orientation of C, the sweep line algorithm to find the closest Steiner point to the apex of each cone C (whenever an obstacle intersects with C) takes $O(h \lg h)$ time. Hence, computing the set of closest Steiner points corresponding to all the cone orientations in $\mathcal{C}$ together take $O(\frac{h}{\sqrt{\epsilon'}} \lg h)$.

The number of nodes in the spanner G is $O(\frac{h}{\sqrt{\epsilon'}})$. These nodes include coreset vertices and at most one closest Steiner point per cone. As each cone introduces at most one edge into G, the number of edges in G is $O(\frac{h}{\sqrt{\epsilon'}})$. Using the Fredman-Tarjan algorithm [13], finding a shortest path between s and t in G takes $O(\frac{h}{\sqrt{\epsilon'}} \lg \frac{h}{\sqrt{\epsilon'}})$ time. Hence, computing the $(1 + \epsilon)$-approximate distance between s and t takes $O(n + \frac{h}{\sqrt{\epsilon'}} \lg \frac{h}{\sqrt{\epsilon'}})$ time. For $\alpha = \frac{\epsilon}{2}$, the value of ϵ' is $O(\epsilon^2)$. Hence, the result stated in the theorem statement.

For the plane sweep, leftmost and rightmost extreme vertices of convex polygons in $\mathcal{P}$ are found in $O(n)$ time. There are $O(h)$ line segments in $\mathcal{T}$, cardinality of Ω is $O(h)$, and $O(h)$ line segment-obstacle pairs (respectively from $\mathcal{T}$ and $\mathcal{P}$) that intersect. The number of event points due to the endpoints in sets L, R, and the endpoints of line segments in $\mathcal{T}$ is $O(h)$. If l and P become non-adjacent along the sweep line, deleting their point of intersection from Q is charged to the event that caused them non-adjacent. The sweep line status is updated if any of these $O(h)$ number of event points occur. Analogous to the analysis provided for line segment intersection [11], our plane sweep algorithm takes $O(n + h \lg h)$ time.

Due to Dobkin et al. [12], determining whether a line segment l in SP_Ω intersects with an obstacle P takes $O(\lg n)$ time, The preprocessing structures corresponding to [12] take $O(n)$ space and they are constructed in $O(n)$ time. Further, replacing every line segment between points of intersection with their respective geodesic shortest paths along the boundaries of obstacles together take $O(n)$ time. ◄

Note that the proof of the above theorem requires us to set the value of α to $\frac{\epsilon}{2}$.

3 Approximate shortest path amid simple polygons

In this section, we extend the approximation method from previous sections to the case of simple (not necessarily convex) polygons. This is accomplished by first decomposing $\mathcal{F}(\mathcal{P})$ into a set of corridors, funnels, hourglasses, and junctions [19, 20, 22]. In the following, we describe these geometric structures, and then we detail our algorithm.

For convenience, we assume a bounding box encloses the polygonal domain $\mathcal{P}$. In the following, we describe a coarser decomposition of $\mathcal{F}(\mathcal{P})$ as compared to the triangulation of $\mathcal{F}(\mathcal{P})$. In specific, this decomposition is used in our algorithm to achieve efficiency. Let $Tri(\mathcal{F})$ denote a triangulation of $\mathcal{F}(\mathcal{P})$. The line segments of $Tri(\mathcal{F})$ that are not the edges of obstacles in $\mathcal{P}$ are referred to as *diagonals*. Let $G(\mathcal{F})$ denote the dual graph of $Tri(\mathcal{F})$, i.e., each node of $G(\mathcal{F})$ corresponds to a triangle of $Tri(\mathcal{F})$ and each edge connects two nodes corresponding to two triangles sharing a diagonal of $Tri(\mathcal{F})$. Based on $G(\mathcal{F})$, we compute a planar 3-regular graph, denoted by G_3 (the degree of every node in G_3 is three), possibly with loops and multi-edges, as follows. First, we remove each degree-one node from $G(\mathcal{F})$ along with its incident edge; repeat this process until no degree-one node remains in the graph. Second, remove every degree-two node from $G(\mathcal{F})$ and replace its two incident edges by a single edge; repeat this process until no degree-two node remains. The resultant graph G_3 is planar, which has $O(h)$ faces, nodes, and edges. Every node of G_3 corresponds to a triangle in $Tri(\mathcal{F})$, called a *junction triangle*. The removal of all junction triangles results in
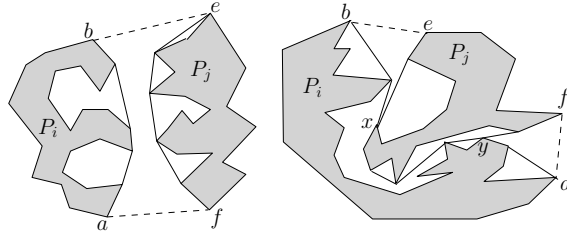
Figure 6 Illustrating an open hourglass (left) and a closed hourglass (right) with a corridor path connecting the apexes x and y of the two funnels. The dashed segments are diagonals.

$O(h)$ *corridors.* The points s and t between which a shortest path needs to be computed are placed in their own degenerate single point corridors. The boundary of each corridor C consists of four parts (see Fig. 6): (1) A boundary portion of an obstacle $P_i \in \mathcal{P}$, from a point a to a point b; (2) a diagonal of a junction triangle from b to a point e on an obstacle $P_j \in \mathcal{P}$ ($P_i = P_j$ is possible); (3) a boundary portion of the obstacle P_j from e to a point f; (4) a diagonal of a junction triangle from f to a. The corridor C is a simple polygon. Let $\tau(a,b)$ (resp., $\tau(e,f)$) be the Euclidean shortest path from a to b (resp., e to f) in C. The region H_C bounded by $\tau(a,b), \tau(e,f), \overline{be}$, and $\overline{fa}$ is called an *hourglass*, which is *open* if $\tau(a,b) \cap \tau(e,f) = \emptyset$ and *closed* otherwise. (Refer Fig. 6.) If H_C is open, then both $\tau(a,b)$ and $\tau(e,f)$ are convex polygonal chains and are called the *sides* of H_C; otherwise, H_C consists of two *funnels* and a path $\tau_C = \tau(a,b) \cap \tau(e,f)$ joining the two apexes of the two funnels, and τ_C is called the *corridor path* of C. Let x and y be the endpoints of π_C. Also, let x be at a shorter distance from b as compared to y. The paths $\tau(b,x), \tau(e,x), \tau(a,y)$, and $\tau(f,y)$ are termed *sides of funnels* of hourglass H_C. We note that these paths are indeed convex polygonal chains. The apieces x and y together is termed a *apex pair* of hourglass H_C. Further, the shortest path between x and y along the boundary of H_C is the *corridor path between apexes* of H_C.

We first give an overview of our algorithm for simple polygonal obstacles. A sketch of $\mathcal{P}$ comprising of a sequence of convex polygonal (core-)chains is computed. Each such core-chain either corresponds to an approximation of a side of an open hourglass or a side of a funnel. If a simple polygon does not participate in any closed corridor, these polygonal chains together form a core-polygon. Similar to the convex polygon case, each such polygonal chain is partitioned into patches. Using these chains, we compute a spanner G. In addition, the following set of edges are included in G: for every closed hourglass H_C and for each obstacle P that participates in H_C, an edge representing the unique shortest path between the two apieces of H_C (as detailed below). After we compute a shortest path p between s and t in the spanner, for every edge $e(r', r'') \in p$, if e is an edge that corresponds to the closed corridor path then we replace e with a shortest path (sequence of edges) between r' and r'' in $\mathcal{F}(\mathcal{P})$. The resultant path is the output of our algorithm. The scheme designed in Agarwal et al. [1] does not appear to extend easily to the case of simple polygons as they use the critical step of computing partitioning planes between pairs of convex polygonal obstacles from $\mathcal{P}$.

For every obstacle $P_j \in \mathcal{P}$, let $\mathcal{R}_j$ be the union of the following: (i) the set comprising of open hourglass sides whose endpoints are incident to P_j, and (ii) the set comprising of sections of funnel sides whose non-apex endpoints incident to P_i. Note that the elements of sets in (i) and (ii) are polygonal convex chains. For every $R \in \mathcal{R}_j$, similar to the case of convex polygonal obstacles, we partition R into patches and the set comprising of the endpoints of these patches is the coreset of R. (For details, refer to Section 2.) For every

$R \in \mathcal{R}_j$, the *core-chain of R* is obtained by joining every two successive vertices that belong to the coreset of R with a line segment while traversing the boundary of R. We construct a spanner $G(V, E)$ that correspond to core-chains of $\mathcal{P}$ using CVDs. For every admissible cone C_p at every vertex p of every core-chain, we consider C_p only if C_p has an intersection with $\mathcal{F}(\mathcal{P})$. While noting that Clarkson's method extends to core-chains defined as above, the shortest path determination algorithm for simple polygons is the same as for the convex polygons described in the previous section except for the following. For each apex pair v'-v'', an edge e is introduced into G between the vertices of G that correspond to v' and v'' with the weight of e equal to the geodesic distance between v' and v'' in the closed hourglass. For a shortest path p between any two nodes of G, for every edge $e \in p$ if both the endpoints of e correspond to an apex pair a'-a'' then we replace p with the shortest path between a' and a'' so that that path contains the corridor path of that closed hourglass; otherwise, as in Lemma 2, we replace the line segment l correspond to e with the sections of l together with the geodesic paths along the boundaries of patches that l intersects. Thus a shortest path between s and t in the spanner G is transformed to a path in the $\mathcal{F}(\mathcal{P})$. In addition, since the distance along the path that contains the corridor path between every pair of apexes is made as the weight of its corresponding edge in the spanner, and due to Lemma 6, the distance along the transformed path is a $(1 + \alpha\epsilon)$-approximation to the distance between s and t amid obstacles in $\mathcal{P}$.

▶ **Lemma 8.** *For a set $\mathcal{P}$ of h pairwise disjoint simple polygons in $\mathbb{R}^2$ and two points $s, t \in \mathcal{F}(\mathcal{P})$, the sketch of $\mathcal{P}$ with cardinality $O(\frac{h}{\sqrt{\alpha\epsilon}})$ suffices to compute a $(1+\alpha\epsilon)$-approximate shortest path between s and t in $\mathcal{F}(\mathcal{P})$.*

Computing hourglasses of $\mathcal{F}(\mathcal{P})$ using [19, 20, 22] and determining the core-chains together takes $O(n + h(\lg h)^{1+\delta} + h \lg n)$ time (where δ is a small positive constant resulting from the triangulation of $\mathcal{F}(\mathcal{P})$ using the algorithm from [3]). Extending the proof of Theorem 7 leads to the following.

▶ **Theorem 9.** *Given a set $\mathcal{P}$ of pairwise disjoint simple polygonal obstacles, two points $s, t \in \mathcal{F}(\mathcal{P})$, and $\epsilon \in (0, 0.6]$, a $(1 + \epsilon)$-approximate geodesic shortest path between s and t is computed in $O(n + h((\lg n) + (\lg h)^{1+\delta} + (\frac{1}{\epsilon} \lg \frac{h}{\epsilon})))$ time. Here, δ is a small positive constant (resulting from the time involved in triangulating $\mathcal{F}(\mathcal{P})$ using [3]).*

Same as in Theorem 7, the proof of this theorem also needs the value of α to be equal to $\frac{\epsilon}{2}$.

4   Two-point approximate distance queries amid convex polygons

We preprocess the given set $\mathcal{P}$ of convex polygons to output the approximate distance between any two query points located in $\mathcal{F}(\mathcal{P})$. Like in the previous section, our preprocessing algorithm relies on [8] and constructs a spanner G. Our query algorithm constructs an auxiliary graph from G. We compute the approximate distance between the two query points using a shortest path finding algorithm in the auxiliary graph.

4.1   Preprocessing

The graph G constructed as part of preprocessing in Section 2.2 is useful in finding an approximate Euclidean shortest path in $\mathcal{F}(\mathcal{P})$ between any two vertices in $\mathcal{P}$. Instead of finding a shortest path between two query nodes in G, to improve the query time complexity, we compute a planar graph $G^{pl}(V, E^{pl})$ from $G(V, E)$ using the result from Chew [6]. Chew's

algorithm finds a set $E^{pl} \subseteq E$ in $O(|V| \lg |V|)$ time so that the distance between any two nodes of G^{pl} is a 2-approximation of the distance between the corresponding nodes in G. We use the algorithm from Kawarabayashi et al. [23] to efficiently answer $(1 + \epsilon)$-approximate distance (length) queries in G^{pl}. More specifically, [23] takes $O(|V|(\lg |V|)^2)$ time to construct a data structure of size $O(|V|)$ so that any distance query is answered in $O((\frac{\lg |V|}{\epsilon})^2)$ time.

▶ **Lemma 10.** *Let G be the spanner computed for the polygonal domain Ω using the algorithm mentioned in Subsection 2.2. Let s and t be two points in $\mathcal{F}(\mathcal{P})$. Let G^{pl} be the planar graph constructed from G using [6]. Further, let $dist_K(s,t)$ be the distance between s and t in G^{pl} computed using the algorithm from [23]. By choosing $\alpha = \frac{\epsilon}{12}$, $dist_{\mathcal{P}}(s,t) \leq dist_K(s,t) \leq (2 + \epsilon)dist_{\mathcal{P}}(s,t)$.*

Proof. From Lemma 4, we know that $dist_\Omega(s,t) \leq dist_{\mathcal{P}}(s,t) \leq (1 + \alpha\epsilon)dist_\Omega(s,t)$. From Lemma 6, we know that $dist_\Omega(s,t) \leq dist_G(s,t) \leq (1 + \sqrt{\alpha\epsilon})dist_\Omega(s,t)$. Let $dist_{G^{pl}}(s,t)$ be the distance in G^{pl} between nodes s and t of G^{pl}. From [6], $dist_G(s,t) \leq dist_{G^{pl}}(s,t) \leq 2\,dist_G(s,t)$. Further, as mentioned above, $dist_{G^{pl}}(s,t) \leq dist_K(s,t) \leq (1 + \alpha\epsilon)dist_{G^{pl}}(s,t)$. As detailed in Lemma 2, algorithm transforms a shortest path between s and t in K to a path p in $\mathcal{F}(\mathcal{P})$. Let $dist_{\mathcal{P}}^p(s,t)$ be the distance along p. From Lemma 2, $dist_{\mathcal{P}}^p(s,t) \leq (1 + \alpha\epsilon)dist_K(s,t)$. Hence, $dist_{\mathcal{P}}^p(s,t) \leq (1 + \alpha\epsilon)dist_K(s,t) \leq (1 + \alpha\epsilon)^2 dist_{G^{pl}}(s,t) \leq 2(1 + \alpha\epsilon)^2 dist_G(s,t) \leq 2(1 + \alpha\epsilon)^2(1 + \sqrt{\alpha\epsilon})dist_\Omega(s,t) \leq 2(1 + \alpha\epsilon)^2(1 + \sqrt{\alpha\epsilon})dist_{\mathcal{P}}(s,t)$. Since p is a path in $\mathcal{F}(\mathcal{P})$, it is immediate to note that $dist_{\mathcal{P}}(s,t) \leq dist_{\mathcal{P}}^p(s,t)$. Therefore, $dist_{\mathcal{P}}(s,t) \leq dist_{\mathcal{P}}^p(s,t) \leq 2(1 + \alpha\epsilon)^2(1 + \sqrt{\alpha\epsilon})dist_{\mathcal{P}}(s,t)$. To achieve $(2 + \epsilon)$-approximation, $(2)(1 + \alpha\epsilon)^2(1 + \sqrt{\alpha\epsilon})$ needs to be less than or equal to $(2 + \epsilon)$. For small values of ϵ ($\epsilon \in (0, 0.7]$), choosing $\alpha = \frac{\epsilon}{12}$ satisfies this inequality. ◀

We note that $\alpha\epsilon$ is $O(\epsilon^2)$. We suppose that there are $O(\frac{1}{\epsilon})$ cones in $\mathcal{C}$, each cone with a cone angle $O(\epsilon)$. It remains to describe data structures that need to be constructed during the preprocessing phase for obtaining the closest vertex of the query point s (resp. t) in a given cone C_s (resp. C_t). To efficiently determine all these $O(\frac{1}{\epsilon})$ neighbors to s and t during query time, we construct a set of $O(\frac{1}{\epsilon})$ CVDs: for every $C \in \mathcal{C}$, one CVD that corresponds to C. The CVDs are constructed similarly to the algorithm given in Subsection 2.2.

▶ **Lemma 11.** *The preprocessing phase takes $O(n + \frac{h}{\epsilon^2}(\lg \frac{h}{\epsilon}) + \frac{h}{\epsilon}(\lg \frac{h}{\epsilon})^2)$ time. The space complexity of the data structures constructed by the end of the preprocessing phase is $O(\frac{h}{\epsilon})$.*

Proof. Computing the sketch Ω from the given $\mathcal{P}$ takes $O(n + \frac{h}{\epsilon})$ time. The number of cones in all the CVDs together is $O(\frac{h}{\epsilon})$. It takes $O(\frac{1}{\epsilon}\frac{h}{\epsilon} \lg \frac{h}{\epsilon''})$ time to compute G which include computing CVDs. Due to [6], computing planar graph G^{pl} with $O(\frac{h}{\epsilon})$ nodes takes $O(\frac{h}{\epsilon} \lg \frac{h}{\epsilon})$ time. Computing space-efficient data structures using [23] takes $O(\frac{h}{\epsilon''}(\lg \frac{h}{\epsilon})^2)$ time. Hence, the preprocessing phase takes $O(n + \frac{h}{\epsilon''} \lg \frac{h}{\epsilon} + \frac{h}{\epsilon}((\lg \frac{h}{\epsilon})^2)$ time. Further, data structures constructed using [23] by the end of preprocessing phase occupy $O(\frac{h}{\epsilon})$ space. The Kirkpatrick's point location [24] data structures for planar point location take $O(\frac{h}{\epsilon})$ space. ◀

4.2 Shortest distance query processing

The query algorithm finds the obstacle-avoiding Euclidean shortest path distance between any two given points $s, t \in \mathcal{F}(\mathcal{P})$. We construct a graph G_{st} from G^{pl}. (The graph G^{pl} is as defined in Subsection 4.1.) For every $C \in \mathcal{C}$, if the point s is located in the cell of a point p of CVD corresponding to C, then we introduce a node corresponding to p into a set V_s. (Essentially, p is the closest visible point in cone $-C_s$ to point s.) Analogously, we define the set V_t of nodes for t in G_{st}. The node set of G_{st} comprises of nodes in $V_s \cup V_t \cup \{s, t\}$.

The edges of this graph are of three kinds: $\{s\} \times V_s, V_s \times V_t$ and $\{t\} \times V_t$. Since there are $O(\frac{1}{\epsilon})$ CVDs, the number of nodes and edges of G_{st} are respectively $O(\frac{1}{\epsilon})$ and $O(\frac{1}{\epsilon^2})$. For every edge (s, s') (resp. (t, t')) with $s' \in V_s$ (resp. $t' \in V_t$), the weight of edge (s, s') (resp. (t, t')) is the Euclidean distance between s and s' (resp. t and t'). For every edge (s', t') with $s' \in V_s$ and $t' \in V_t$, the weight of (s', t') is the $(2 + \epsilon)$-approximate distance between s' and t'. These weights are obtained from the data structures maintained as in [23]. We use Fredman-Tarjan algorithm [13] to find a shortest path between s and t in G_{st}. From the above, this distance is a $(2 + \epsilon)$-approximate distance from s to t amid convex polygons in $\mathcal{P}$.

▶ **Theorem 12.** *Given a set $\mathcal{P}$ of h pairwise disjoint convex polygonal obstacles in plane defined with n vertices and $\epsilon \in (0, 0.6]$, the polygons in $\mathcal{P}$ are preprocessed in $O(n + \frac{h}{\epsilon^2}(\lg \frac{h}{\epsilon}) + \frac{h}{\epsilon}(\lg \frac{h}{\epsilon})^2)$ time to construct data structures of size $O(\frac{h}{\epsilon})$ for answering two point $(2 + \epsilon)$-approximate distance query between any two given points belonging to $\mathcal{F}(\mathcal{P})$ in $O(\frac{1}{\epsilon^6}(\lg \frac{h}{\epsilon})^2)$ time.*

References

1   P. K. Agarwal, R. Sharathkumar, and H. Yu. Approximate Euclidean shortest paths amid convex obstacles. In *Proceedings of Symposium on Discrete Algorithms*, pages 283–292, 2009.

2   S. R. Arikati, D. Z. Chen, L. P. Chew, G. Das, M. H. M. Smid, and C. D. Zaroliagis. Planar spanners and approximate shortest path queries among obstacles in the plane. In *Proceedings of European Symposium on Algorithms*, pages 514–528, 1996.

3   R. Bar-Yehuda and B. Chazelle. Triangulating disjoint Jordan chains. *International Journal of Computational Geometry & Applications*, 4(4):475–481, 1994.

4   D. Z. Chen. On the all-pairs Euclidean short path problem. In *Proceedings of Symposium on Discrete Algorithms*, pages 292–301, 1995.

5   D. Z. Chen and H. Wang. Computing shortest paths among curved obstacles in the plane. *ACM Transactions on Algorithms*, 11(4):26:1–26:46, 2015.

6   L. P. Chew. There are planar graphs almost as good as the complete graph. *Journal of Computer and System Sciences*, 39(2):205–219, 1989.

7   Y.-J. Chiang and J. S. B. Mitchell. Two-point Euclidean shortest path queries in the plane. In *Proceedings of Symposium on Discrete Algorithms*, pages 215–224, 1999.

8   K. L. Clarkson. Approximation algorithms for shortest path motion planning. In *Proceedings of Symposium on Theory of Computing*, pages 56–65, 1987.

9   T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms.* The MIT Press, 2009.

10   Toth C. D., O'Rourke J., and Goodman J. E. *Handbook of discrete and computational geometry.* CRC Press, 3rd ed. edition, 2017.

11   M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: algorithms and applications.* Springer-Verlag, 3rd ed. edition, 2008.

12   D. P. Dobkin and David G. Kirkpatrick. Fast detection of polyhedral intersection. *Theoretical Computer Science*, 27:241–253, 1983.

13   M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of ACM*, 34(3):596–615, 1987.

14   S. K. Ghosh. *Visibility algorithms in the plane.* Cambridge University Press, New York, USA, 2007.

15   L. J. Guibas and J. Hershberger. Optimal shortest path queries in a simple polygon. *Journal of Computer and System Sciences*, 39(2):126–152, 1989.

16   H. Guo, A. Maheshwari, and J-R. Sack. Shortest path queries in polygonal domains. In *Proceedings of Algorithmic Aspects in Information and Management*, pages 200–211, 2008.

17   J. Hershberger and S. Suri. An optimal algorithm for Euclidean shortest paths in the plane. *SIAM Journal on Computing*, 28(6):2215–2256, 1999.

18   R. Inkulu, S. Kapoor, and S. N. Maheshwari. A near optimal algorithm for finding Euclidean shortest path in polygonal domain. *CoRR*, abs/1011.6481, 2010. `arXiv:1011.6481`.

19   S. Kapoor. Efficient computation of geodesic shortest paths. In *Proceedings of Symposium on Theory of Computing*, pages 770–779, 1999.

20   S. Kapoor and S. N. Maheshwari. Efficent algorithms for Euclidean shortest path and visibility problems with polygonal obstacles. In *Proceedings of Symposium on Computational Geometry*, pages 172–182, 1988.

21   S. Kapoor and S. N. Maheshwari. Efficiently constructing the visibility graph of a simple polygon with obstacles. *SIAM Jounral on Computing*, 30(3):847–871, 2000.

22   S. Kapoor, S. N. Maheshwari, and J. S. B. Mitchell. An efficient algorithm for Euclidean shortest paths among polygonal obstacles in the plane. *Discrete & Computational Geometry*, 18(4):377–383, 1997.

23   K. Kawarabayashi, P. N. Klein, and C. Sommer. Linear-space approximate distance oracles for planar, bounded-genus and minor-free graphs. In *Proceedings of Colloquium on Automata, Languages and Programming*, pages 135–146, 2011.

24   D. G. Kirkpatrick. Optimal search in planar subdivisions. *SIAM Journal on Computing*, 12(1):28–35, 1983.

25   J. Kleinberg and E. Tardos. *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., 2005.

26   J. S. B. Mitchell. Shortest paths among obstacles in the plane. *International Journal of Computational Geometry & Applications*, 6(3):309–332, 1996.

27   J. S. B. Mitchell. Geometric shortest paths and network optimization. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 633–701. North-Holland, 2000.

28   S. Sen. Approximating shortest paths in graphs. In *Proceedings of Workshop on Algorithms and Computation*, pages 32–43, 2009.

29   E. Welzl. Constructing the visibility graph for n-line segments in $O(n^2)$ time. *Information Processing Letters*, 20(4):167–171, 1985.

30   A. C. Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal on Computing*, 11(4):721–736, 1982.

A   Appendix

Comparison with previous results is depicted in Table 1.

In Chiang and Mitchell [7], h_s (resp. h_t) is the number of vertices visible from s (resp. t). Both the Chen [4] and Arikati et al. [2] output a shortest path in additional $O(L)$ time, where L is the number of edges of the output path. The r in Arikati et al. [2] is an arbitrary integer such that $1 \le r \le n$.

Table 1 Comparison with previous results.

	Preprocessing time	Space	Query time	Time	Stretch	Comment
Our results	-	-	-	$O(n + h((\lg n) + (\lg h)^{1+\delta} + \frac{1}{\epsilon}\lg\frac{h}{\epsilon}))$	$1+\epsilon$	non-convex
	-	-	-	$O(n + \frac{h}{\epsilon}\lg\frac{h}{\epsilon})$	$1+\epsilon$	convex
	$O(n + \frac{h}{\epsilon^2}(\lg\frac{h}{\epsilon}) + \frac{h}{\epsilon}(\lg\frac{h}{\epsilon})^2)$	$O(\frac{h}{\epsilon})$	$O(\frac{1}{\epsilon^6}(\lg\frac{h}{\epsilon})^2)$	-	$2+\epsilon$	convex
Agarwal et al. [1]	-	-	-	$O(n + \frac{h}{\sqrt{\epsilon}}\lg(\frac{h}{\epsilon}))$	$1+\epsilon$	convex
Chiang & Mitchell [7]	-	$O(n^{5+\epsilon})$	$o(n)$	-	optimal	non-convex
	-	$O(n^{5+10\delta+\epsilon})$	$O(n^{1-\delta}\lg n)$	-	optimal	non-convex
	-	$O(n^{10}\lg n)$	$O((\lg n)^2)$	-	optimal	non-convex
	-	$O(n^{11})$	$O(\lg n)$	-	optimal	non-convex
	-	$O(n^5)$	$O(\lg n + \min h_s, h_t)$	-	optimal	non-convex
	-	$O(n + h^5)$	$O(h\lg n)$	-	optimal	non-convex
Chen [4]	-	$O(n\lg n + \frac{n}{\epsilon})$	$O(\frac{\lg n}{\epsilon} + \frac{1}{\epsilon^2})$	-	$6+\epsilon$	non-convex
Arikati et al. [2]	$O(\frac{n^2}{\sqrt{r}})$	$O(\frac{n^2}{\sqrt{r}})$	$O(\lg n + \sqrt{r})$	-	$\sqrt{2}+\epsilon$	non-convex
	$O(n\lg n)$	$O(n)$	$O(n)$	-	$\sqrt{2}+\epsilon$	non-convex
	$O(n^{3/2})$	$O(n^{3/2})$	$O(\lg n)$	-	$2\sqrt{2}+\epsilon$	non-convex
	$O(\frac{n^{3/2}}{\sqrt{\lg n}})$	$O(n\lg n)$	$O(\lg n)$	-	$3\sqrt{2}+\epsilon$	non-convex