Approximate Euclidean Shortest Paths amid Polygonal Obstacles

R. Inkulu† † Sanjiv Kapoor‡

Abstract

Given a set $P$ of non-intersecting polygonal obstacles in $\mathbb{R}^2$ defined with $n$ vertices, we compute a sketch $\Omega$ of $P$ whose size is independent of $n$. We utilize $\Omega$ to devise an algorithm to compute a $(1 + \epsilon)$-approximate Euclidean shortest path between two points given with the description of $P$. When $P$ comprises of convex polygonal obstacles, we devise a $(2 + \epsilon)$-approximation algorithm to efficiently answer two-point Euclidean distance queries.

1 Introduction

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 the free space $F(\mathcal{P})$ defined by the closure of $\mathbb{R}^2$ sans the union of interior of all the simple polygons in $\mathcal{P}$, the Euclidean shortest path finding problem asks to compute a shortest path between $s$ and $t$ that lies in $F(\mathcal{P})$. This problem is well-known in computational geometry community. Mitchell [22] provides an extensive survey of research accomplished in determining shortest paths in polygonal and polytope domains to that date. In the following, we assume that $n$ vertices together define $h$ obstacles of $\mathcal{P}$.

The polygonal domain $\mathcal{P}$ is given as input, a priori. The three variants of the problem include: (i) both $s$ and $t$ are given as input with $\mathcal{P}$, (ii) only $s$ is given 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. In a type (ii) problem, the preprocessing phase 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, $\mathcal{P}$ 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 type (i) or type (ii) problem, there are two fundamentally different approaches: the visibility graph method (see Ghosh [10] for both the survey and details of various visibility algorithms) and the wavefront method. The visibility graph method [23] is based on constructing a graph $G$ 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 a quadratic time complexity barrier. In the wavefront based approach [12] [21] [16] [13], a wavefront is expanded from $s$ till it reaches $t$. The wavefront method typically constructs a shortest path map with respect to $s$ so that for any query point $t$, a shortest path from
The two-point shortest path query problem within a given simple polygon was addressed by Guibas and Hershberger [11]. Their result preprocesses the given simple polygon in time $O(n)$ and constructs a data structure of size $O(n)$ and answers two-point shortest path distance queries in time $O(\lg n)$. The exact two-point shortest path queries in polygonal domain were explored by Chiang and Mitchell [5]. This result primarily devises two algorithms: one to construct data structures of size $O(n^5)$ that facilitates in answering distance queries in $O(h + \lg n)$ time; the other algorithm builds data structures of size $O(n + h^2)$ and yields $O(h \lg n)$ query 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. [7] preprocesses $\mathcal{F}(\mathcal{P})$ in $O(n^2 \lg n)$ time to construct data structure of size $O(n^2)$ to answer two-point Euclidean 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 [6]. Chen [3] used the techniques from [6] in constructing a data structure of size $O(n \lg n + \frac{n}{r})$ in $o(n^{3/2} + O(\frac{\sqrt{n}}{\epsilon} \lg \frac{n}{\epsilon}))$ time to support $O(6 + \epsilon)$-approximate two-point distance queries in $O(\frac{h}{\epsilon^2} + \frac{1}{\epsilon^6})$ time. Arikati et al. [2] gave a family of results with trade-offs among the approximation factor, preprocessing time, storage space and the query time. Agarwal et al. [11] computes an approximate shortest path in $O(n + \frac{h}{\epsilon^3} \lg \frac{n}{\epsilon})$ time when the obstacles are convex.

All through this document, to distinguish graph vertices from the vertices of the polygonal domain, we refer to vertices of graph as nodes. The Euclidean distance between any two points $p$ and $q$ is denoted with $\|pq\|$. The obstacle avoiding geodesic distance between any two points $p,q$ amid a set $Q$ of obstacles is denoted with $distQ(p,q)$. We denote the convex hull of a set $R$ of points or a simple polygon $R$ with $CH(R)$. For any set of polygonal obstacles $Q$, the free space resultant from the closure of $\mathbb{R}^2$ sans the union of interior of all the polygons in $Q$ is denoted with $\mathcal{F}(Q)$. Let $r'$ and $r''$ be two rays with origin at $p$, and respectively make $\theta'$ and $\theta''$ counterclockwise angles with the positive $x$-axis in a coordinate system. Let $\overrightarrow{v_1}$ and $\overrightarrow{v_2}$ be the respective unit vectors corresponding to rays $r'$ and $r''$. 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 vectors $\overrightarrow{v_1}$ and $\overrightarrow{v_2}$. When the rays are obvious 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 contribution**

We compute a sketch $\Omega$ comprising of simple polygonal obstacles from the set $\mathcal{P}$ of input simple polygonal obstacles. The sketch $\Omega$ has $h$ simple polygonal obstacles: for $1 \leq i \leq h$, simple polygon $P_i \in \mathcal{P}$ is approximated with another simple polygon $Q_i \in \Omega$ such that $Q_i \subseteq CH(P_i)$. In computing sketch $\Omega$ of $\mathcal{P}$, we identify coreset $S_i$ of vertices of each polygon $P_i \in \mathcal{P}$. Further, we form a corepolygon $Q_i \in \Omega$, which is again a simple polygon, by joining every successive pairs of vertices of $S_i$ that occur while traversing $P_i$ in counterclockwise order with a line segment. Note that when $P_i$ is convex, the corresponding corepolygon $Q_i$ obtained through this procedure is both convex as well as $Q_i \subseteq P_i$. Like in [11], combinatorial complexity of $\Omega$ is independent of the number of vertices defining $\mathcal{P}$. For two points $s,t \in \mathcal{F}(\mathcal{P})$, we compute an approximate Euclidean path between $s$ and $t$ in $\mathcal{F}(\Omega)$ using an algorithm that is a variant of [6]. From this path, we compute
another path in $\mathcal{F}(\mathcal{P})$ and show that this new path is a $(1 + \epsilon)$-approximate shortest path between $s$ and $t$ amid polygons in $\mathcal{P}$. The main contributions of this paper are summarized below:

* When $\mathcal{P}$ comprises of convex polygonal obstacles, we preprocess those obstacles in $O(n + \frac{h}{\epsilon^2} \log \frac{h}{\epsilon^2} + \frac{h}{\epsilon^2} \log^2 \frac{h}{\epsilon^2})$ time to construct data structures of size $O(\frac{h}{\epsilon^2})$ to answer two-point $(2 + \epsilon)$-approximate geodesic Euclidean distance (length) queries in $O(\frac{1}{(\epsilon^2)^2} \log^2 \frac{h}{\epsilon^2})$ time, where $\epsilon'' = (\frac{2 + \epsilon}{2})^{1/3} - 1$.

* As part of devising the above algorithm (i.e., again when $\mathcal{P}$ comprises of convex polygonal obstacles), we devise an algorithm to compute a $(1 + \epsilon)$-approximate Euclidean distance between two points $s, t \in \mathcal{F}(\mathcal{P})$, given the description of $\mathcal{P}$, in $O(n + \frac{h}{\epsilon^2} \log \frac{h}{\epsilon^2})$ time, where $\epsilon' = \sqrt{1 + \epsilon} - 1$. Further, a $(1 + \epsilon)$-approximate Euclidean shortest path is computed in additional $O(h \log n)$ time.

* Further, we extend the above algorithm for computing a $(1 + \epsilon)$-approximate Euclidean shortest path between $s$ and $t$ in $O(n + h((\log n) + (\log h)^{1+\delta} + \frac{1}{\sqrt{\epsilon'}} \log \frac{h}{\epsilon'}))$ time when $\mathcal{P}$ comprises of simple polygons. Here $\epsilon'$ equals to $\sqrt{1 + \epsilon} - 1$, and $\delta$ is a small positive constant due to time involved in triangulating $\mathcal{F}(\mathcal{P})$ using [15].

In addition, our algorithm to compute the coreset of simple polygons and obtaining a sketch of $\mathcal{P}$ may be of independent interest in devising efficient approximation algorithms to other geometric optimization problems.

As mentioned above, Agarwal et al. [1] devises a $(1 + \epsilon)$-approximation algorithm to compute a $s$-$t$ Euclidean shortest path in $O(n + \frac{h}{\epsilon^2} \log \frac{h}{\epsilon^2})$ time when obstacles in $\mathcal{P}$ are convex. Our algorithm is applicable even when polygonal obstacles in $\mathcal{P}$ are non-convex. Our approach of computing coresets is quite different from [1]. The key differences between our algorithm and [1] are detailed below. For two-point shortest paths, Chiang and Mitchell [5] outputs an optimal shortest path; our result answers approximate two-point distance queries with time-space trade-offs with respect to [5]. Moreover, our result trades-off with approximation algorithms devised in [2, 3].

In $\mathbb{R}^2$, our algorithm is very different from [1]. Let the polygonal domain $\mathcal{P}$ be 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 $Q_i \subset P_i$ whereas in [1], $P_i \subset Q_i$.

Let the new polygonal domain $\Omega$ be defined with simple polygons $Q_1, Q_2, \ldots, Q_h$. Unlike [1], in computing $\Omega$, our algorithm does not require using plane-sweep algorithm to find pairwise vertically visible simple polygons of $\mathcal{P}$. Our algorithm partitions the boundary of each convex polygon $P$ into a set of contiguous patches so that the angle subtended by any two points belonging to same patch is upper bounded as a function of $\epsilon$. For any two points $s, t \in \mathcal{F}(\mathcal{P})$, our algorithm guarantees that $\text{dist}_{\Omega}(s, t) \leq \text{dist}_{\mathcal{P}}(s, t) \leq (1 + \epsilon)\text{dist}_{\Omega}(s, t)$. For any two points $s, t \in \mathcal{F}(\Omega)$, algorithm in [1] guarantees that $\text{dist}_{\Omega}(s, t) \leq (1 + \epsilon)\text{dist}(s, t)$. To find a shortest path amid $\Omega$, our algorithm does not use algorithm from [12]. Instead, our algorithm achieves the said approximation using the spanner constructed from cone Voronoi diagrams (CVDs) [6]. Apart from computing sketch $\Omega$ of $\mathcal{P}$, as compared with [6], number of cones per obstacle that participate in computing CVDs amid $\mathcal{F}(\Omega)$ is further optimized to achieve the above mentioned time and space complexities. This is achieved by exploiting the convexity of obstacles together with the properties of shortest paths amid convex obstacles. Further, in our algorithm, for any maximal line segment with endpoints $r', r''$ along the computed (approximate) shortest path $p$ amid obstacles in $\Omega$, if $r'r''$ lies in $P_i - Q_i$
for any $1 \leq i \leq h$, then before outputting $p$ we replace line segment $r'r''$ with geodesic shortest path between $r'$ and $r''$ in $\mathcal{F}(P)$ i.e., with a shortest path between $r'$ and $r''$ along the boundary of $P_i$.

More importantly, the definition of coresets and corepolygons of convex polygonal obstacles is extended to simple polygons using the decomposition of $\mathcal{F}(P)$ into hourglasses \cite{14, 18, 16}. And, the sketch $\Omega$ is defined to comprise simple polygons, like in the convex polygonal case each $Q_i \in \Omega$ correspond to an obstacle $P_i \in \mathcal{P}$ and is obtained from the coreset $S_i$ of $P_i$. The scheme designed in Agarwal et al. \cite{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 $\mathcal{P}$.

The algorithm for single-shot approximate shortest path computation when obstacles are convex polygonal is described in Section 2. Section 3 describes algorithm for computing an approximate Euclidean shortest path amid simple polygonal obstacles. The preprocess-query algorithm to answer approximate Euclidean distance queries is described in Section 4. The conclusions are in Section 5.

2 Approximate shortest path amid convex polygons

In this Section, we suppose that every polygon belonging to $\mathcal{P}$ is convex. We use the following notation from Yao \cite{24}. Let $C$ be the set of cones with disjoint interiors partitioning $\mathbb{R}^2$, where each cone has an apex at the origin and the cone angle is upper bounded by a value that is a function of $\epsilon$. Each cone that we refer in this paper is a translated copy of some cone in $C$. When a cone $C \in C$ is translated to have apex at a point $p$, the translated cone is denoted with $C_p$. We show that $O(\frac{h}{\sqrt{\alpha \epsilon}})$ vertices ($\alpha$ defined later), selected from the set of vertices defining $\mathcal{P}$, together with the select set of cones introduced at these vertices, suffice to compute an approximate shortest path between any two points in $\mathcal{F}(P)$ with the desired accuracy.

2.1 Computing a sketch of $\mathcal{P}$

First, we detail an algorithm to compute a coreset of $\mathcal{P}$. Let $e_j,e_{j+1},\ldots,e_k$ be a sequence $C$ of edges when the boundary of a convex polygon $P_i$ is traversed in counterclockwise order such that for every $j \leq l \leq k - 1$, vertex $v_l$ is common to edges $e_l$ and $e_{l+1}$. The absolute value of difference in angle made by $e_j$ and $e_k$ with the positive $x$-axis is defined as the angle subtended by $C$. Let $\Pi_i$ be a partition of the boundary of a convex polygon $P_i$ into a collection of $\lceil \frac{2\pi}{\alpha \epsilon} \rceil$ contiguous sections, patches, such that the angle subtended by any contiguous section is upper bounded by $\sqrt{\alpha \epsilon}$, with $\alpha$ defined in terms of $\epsilon$ later.

**Lemma 2.1** For any two points $p$ and $q$ that belong to any patch $R \in \Pi_j$, the geodesic Euclidean distance between $p$ and $q$ along $R$ is upper bounded by $(1 + \alpha \epsilon) ||pq||$.

**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 $q$ belong to the same patch, the angle between $cp$ and $cq$ is upper bounded by $\sqrt{\alpha \epsilon}$. The geodesic distance between $p$ and $q$ along $R$ is upper bounded by $\frac{||pq|| \sqrt{\alpha \epsilon}}{\sin \sqrt{\alpha \epsilon}} \leq \frac{||pq|| \sqrt{\alpha \epsilon}}{\sqrt{\alpha \epsilon - \frac{\alpha \epsilon}{6}}} \leq \frac{||pq||}{(1 - \frac{1}{6})} \leq (1 + \frac{\alpha \epsilon}{6}) ||pq|| \leq (1 + \alpha \epsilon) ||pq||$. \qquad $\Box$

For each obstacle $P_i$, the coreset $S_i$ of $P_i$ comprises of two vertices chosen from each patch in $\Pi_i$. In particular, for any patch $\pi \in \Pi_i$ defined with the sequence $v_j,v_{j+1},\ldots,v_{k-1},v_k$ of vertices along an obstacle $P_i$, both the vertices $v_j$ and $v_k$ belong to the coreset $S_i$ of $P_i$. The coreset $S$ of $\mathcal{P}$ is then simply $\bigcup_i S_i$. 

4
Observation 1 The size of coreset \( S \) of \( \mathcal{P} \) is \( O\left(\frac{1}{\sqrt{n}}\right) \).

Noting that the complexity of \( CH(S_i) \) is upper bounded by the complexity of \( P_i \), to achieve the efficiency, for every \( 1 \leq i \leq h \), our algorithm uses corepolygon \( Q_i = CH(S_i) \) in place of \( P_i \).

Let \( \Omega \) be a set comprising of corepolygons corresponding to each of the polygons in \( \mathcal{P} \). Then \( \Omega \) is the sketch of \( \mathcal{P} \). The following Lemmas show that we can achieve a \((1 + \epsilon)\)-approximation amid \( \mathcal{P} \) while using the sketch of \( \mathcal{P} \).

Lemma 2.2 Let \( v', v'' \) be any two vertices of obstacles in \( \Omega \). Then, \( \text{dist}_{\mathcal{P}}(v', v'') \leq (1 + \alpha \epsilon) \text{dist}_{\Omega}(v', v''). \)

Proof: Let \( v_1 \) and \( v_2 \) be any two successive vertices along a shortest path between \( v' \) and \( v'' \) amid \( \Omega \). Let \( \mathcal{O} \) be the set of obstacles intersected by line segment \( v_1v_2 \) in \( \mathcal{P} \). Suppose that \( v_1 \) and \( v_2 \) belong to obstacles \( P_j \) and \( P_k \) respectively. Since the line segment \( v_1v_2 \) does not intersect with the interior of the \( CH(S_j) \) or \( CH(S_k) \), it intersects with at most one patch belonging to set \( \Pi_j \) of patches of \( P_j \) and at most one patch belonging to set \( \Pi_k \) of patches of \( P_k \). (Refer Fig. 1) Let the line segment \( v_1v_2 \) intersect with a patch \( R \in \Pi_j \) at a points \( v_1 \) and \( r \). Then from Lemma 2.1 the geodesic distance between \( v_1 \) and \( r \) along \( R \) is upper bounded by \((1 + \alpha \epsilon) \parallel v_1r \parallel \). Analogously, let the line segment \( v_1v_2 \) intersect with a patch \( R' \in \Pi_k \) at points \( v_2 \) and \( r' \). Then the geodesic distance from \( v_2 \) and \( r' \) is upper bounded by \((1 + \alpha \epsilon) \parallel v_2r' \parallel \). For any convex polygonal obstacle \( P_i \) in \( \mathcal{O} \) distinct from \( P_j \) and \( P_k \), let \( p', p'' \) be the points of intersection of \( v_1v_2 \) with the boundary of \( P_i \). Since the line segment \( v_1v_2 \) does not intersect with the interior of the convex hull of coreset corresponding to \( P_i \), both \( p' \) and \( p'' \) belong to the same patch, say \( R'' \in \Pi_i \). Then again from Lemma 2.1, the geodesic distance from \( p' \) and \( p'' \) along patch \( R'' \) is upper bounded by \((1 + \alpha \epsilon) \parallel p'p'' \parallel \).

Figure 1: A line segment \( v_1v_2 \) of a shortest path amid \( \Omega \) intersecting three patches belonging to obstacles in \( \mathcal{P} \)

Let \( \pi_1, \pi_2, \ldots, \pi_k \) be the set \( \Pi \) of patches intersected by \( v_1v_2 \), and let \( p'_i, p''_i \) be the points of intersections of \( v_1v_2 \) with any patch \( \pi_i \in \Pi \). Then \( \sum_{i=1}^{k} \text{dist}_{\mathcal{P}}(p'_i, p''_i) \) added with \( \sum_{i=1}^{k-1} \parallel p''_i p'_{i+1} \parallel \) is upper bounded by \((1 + \alpha \epsilon) \parallel v_1v_2 \parallel \). Let \( v_1, \ldots, v_r \) be the vertices of \( \mathcal{P} \) that occur in that order along a geodesic shortest path between vertices \( v', v'' \in \mathcal{P} \) amid \( \Omega \). Then \( \text{dist}_{\mathcal{P}}(v_1, v_r) = \sum_{i=1}^{r-1} \text{dist}_{\mathcal{P}}(v_i, v_{i+1}) \leq (1 + \alpha \epsilon) \sum_{i=1}^{r-1} \text{dist}_{\Omega}(v_i, v_{i+1}) \).

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

Lemma 2.3 For any two vertices \( v', v'' \) of \( \mathcal{P} \), \( \text{dist}_{\Omega}(v', v'') \leq \text{dist}_{\mathcal{P}}(v', v'') \).

Thus we ensure that for source and destination vertices \( s, t \in \mathcal{P} \), shortest path computed amid \( \Omega \) achieves \((1 + \alpha \epsilon)\)-approximation.

Lemma 2.4 Let \( \mathcal{P} \) be a collection of \( h \) convex polygons in \( \mathbb{R}^2 \) with \( n \) vertices and let \( s \) and \( t \) be two points in \( \mathcal{F}(\mathcal{P}) \). Then the sketch \( \mathcal{S} \) of \( \mathcal{P} \) with cardinality \( O\left(\frac{h}{\sqrt{n \alpha \epsilon}}\right) \) suffices to compute a \((1 + \alpha \epsilon)\)-approximate shortest path between \( s \) and \( t \) in \( \mathcal{F}(\mathcal{P}) \).
2.2 Computing an approximate shortest path using the sketch $\Omega$ of $\mathcal{P}$

Our algorithm relies on [6]; hence, we give a brief overview of the algorithm from [6] to construct a Euclidean spanner $G(V,E)$ from the given set of obstacles $\mathcal{P}$. Noting that the endpoints of segments of a shortest path in $\mathbb{R}^2$ comprises of only vertices of $\mathcal{P}$, 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 apexed at origin that together partition $\mathcal{F}(\mathcal{P})$. Let $C \in \mathcal{C}$ be a cone with orientation $\theta$ and let $C' \in \mathcal{C}$ be the cone with orientation $-\theta$. A cone Voronoi diagram $CVD$ is constructed corresponding to each cone $C \in \mathcal{C}$ such that for a cone $C \in \mathcal{C}$ and set $K$ of points, we let $CVD(C,K)$ be a partition of $\mathcal{F}(\mathcal{P})$ where for every point $p \in K$ there is an associated region $R_p \subseteq \mathcal{F}(\mathcal{P})$, $R_p \in CVD(C,K)$. $R_p$ is defined using the cones in $\mathcal{C}$. Indeed, the region $R_p$ comprises of all points $q \in \mathcal{F}(\mathcal{P})$ such that $p$ is the closest vertex in $C'_q$ among points in $K$. For a given cone, $C_v$, let $V'$ be the set of vertices of $\mathcal{P}$ that are visible from $v$ and that lie within the cone $C_v$. The vertex in $V'$ that is closest to $v$, say $v'$, is said to be the closest vertex in $C_v$ to $v$. For every vertex $v$ of $\mathcal{P}$ and for every cone $C_v$, an edge $e$ joining $v$ and a closest vertex in $C_v$ to $v$, say $v'$, is introduced in $E$ with its weight equal to the Euclidean distance between $v$ and $v'$. Apart from these, no additional edges are added to $E$. The result in [6] proves that if $d$ is the obstacle avoiding geodesic Euclidean 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 $(1+\epsilon)d$. Further, [6] computes $CVD(C,K)$ using plane-sweep in $O(|K| \log |K|)$ time; and, well-known planar point location structures facilitate locating any point $q$ in any $CVD$ while help in designing shortest path query algorithms.

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 [6]. 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 $v_0, v_1, \ldots, v_k$ be vertices such that $v_1, \ldots, v_{k-1}$ belong to an obstacle $\mathcal{P}$ and $v_0, v_k$ belong to obstacles $\mathcal{P}'$ and $\mathcal{P}''$ respectively, for $\mathcal{P} \neq \mathcal{P}' \neq \mathcal{P}''$. Also, let $v_0v_1 \ldots v_k$ be a subpath $\pi$ of a shortest path. Since $\mathcal{P}$ is a convex polygon, $\angle v_jv_{j+1}v_{j+2}$ interior to $\mathcal{P}$ is less than $\pi$ radians for every $j \in [0, k-2]$, i.e. the subpath is convex w.r.t. $\mathcal{P}$ at each of the vertices $v_1, v_2, \ldots, v_{k-1}$.

![Figure 2: Illustrating an admissible cone $C_v$ incident to a coreset vertex $v$ of an obstacle](image)

Let $v$ be a vertex of $\mathcal{P}$ that belongs to coreset $\mathcal{S}_i$ 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 $(vv', -vv'')$ and let $C''$ be the cone defined by the pair of rays $(vv'', -vv')$. For a coreset vertex $v \in \mathcal{S}_i$, 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. 2.) Note that any shortest path is convex at $v$ with respect to $P_i$. Thus if $q$ is any point in $\mathcal{F}(\mathcal{P})$ such that $q$ is not visible to $p$ amid $\mathcal{P}$ and a shortest path between $p$ and $q$ passes through vertex $v$ of $P_i$, then there exists a shortest path from $p$ to $q$ such that one of its line segment lies in $C'$ and another line segment of that path lies in $C''$. Hence, in computing geodesic shortest path amid $\mathcal{P}$, it suffices to consider admissible cones at 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 domain $\Omega$ as well. For any two points $p_1$ and $p_2$ in $F(\Omega)$, suppose $p_1$ and $p_2$ are not visible to each other. Consider any shortest path $\pi$ between $p_1$ and $p_2$. For any line segment $ab$ in $\pi$, $ab$ is either an edge of $\Omega$ or it is a tangent to an obstacle $O$. In the latter case, $ab$ belongs to an admissible cone of $O$.

The following Lemma upper bounds the number of cones introduced.

**Lemma 2.5** The number of cones introduced at all the obstacles of $\Omega$ together is $O(\frac{h}{\sqrt{\alpha \epsilon}})$.

**Proof:** Let $\vec{r}$ be a ray with origin of the coordinate axes as its endpoint. (See Fig. 3) For any two distinct vertices $v'$ and $v''$ of a convex polygon $P$, let $\vec{r}_{v'}$ be the ray parallel to $\vec{r}$ with origin at $v'$ and pointing in the same direction as $\vec{r}$ and let $\vec{r}_{v''}$ be the ray parallel to $r$ with origin at $v''$ and point in the same direction as $r$. Also, let $v'_1$ (precede $v'$) (resp. $v''_1$ precede $v''$) and $v'_2$ succeed $v'$ (resp. $v''_2$ succeed $v''$) while traversing $P$ in counterclockwise order. Since $P$ is a convex polygon, if $\vec{r}_{v'}$ belong to cone defined by $\vec{v'}_1\vec{v'}_2$ and $\vec{v'}_1\vec{v''}$ then $\vec{r}_{v''}$ is guaranteed to not belong to cone defined by $\vec{v''}_1\vec{v''}_2$ and $\vec{v''}_1\vec{v''}$.

![Figure 3: Illustrating that a ray parallel to $r$ can exist in only one admissible cone per obstacle](image)

Extending this argument, if a cone $C_{v'}$ is admissible 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 $O(\frac{h}{\sqrt{\alpha \epsilon}})$. $\square$

![Figure 4: Illustrating an edge of the spanner](image)

Next, we describe computing the spanner $G(V = S \cup S', E)$. The set $S$ comprises of nodes corresponding to coreset $S$. For every $v \in S$ and for every admissible cone $C_v$, let $V'$ be the set of points on the boundaries of obstacles of $\Omega$ that are visible from $v$ and lie in $C_v$. (See Fig. 4) The point in $V'$ that is closest to $v$, say $p$, the closest Steiner point in $C_v$ to $v$ is found and $p$ is added to $S'$. An edge $e$ between $v$ and $p$ is introduced in $E$ while the Euclidean distance from $v$ and $v'$ 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 lie 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 Euclidean
distance from $p$ to $v'$ (resp. $p$ and $v''$) is set as the weight of $e'$ (resp. $e''$) in $G$. The set $S'$ comprises of all such Steiner points. Note that both $|V|$ and $|E|$ are $O(\frac{1}{\sqrt{\alpha}})$.

**Lemma 2.6** Let $G$ be the spanner constructed from $\Omega$. Let $\text{dist}_G(p',p'')$ be the distance between $p'$ and $p''$ in $G$. Then for any two points $s,t \in F(\Omega)$, $\text{dist}_\Omega(s,t) \leq \text{dist}_G(s,t) \leq (1 + \alpha)\text{dist}_\Omega(s,t)$.

**Proof:** Theorem 2.5 of [6] 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 $\psi$ yields $\psi \leq \sqrt{\alpha\epsilon}$. Hence, the choice of the cone angle of cones and the cardinality of $C$ in Lemma 2.5.

We claim that introducing a subset of cones (admissible cones) rather than all the cones as used in [6] does not affect the correctness. Let $p$ and $q$ be vertices of convex polygons $C_p$ and $C_q$ respectively. Suppose $pq$ is a line segment belonging to a shortest path in the spanner computed in [6]. If $pq$ is a tangent to convex polygonal obstacle $P_i$ at $p$, then $pq$ belong to an admissible cone $C_p$ at $p$. Similarly, if $pq$ is a tangent to convex polygonal obstacle $P_j$ at $q$, then $pq$ belong to an admissible cone at $q$. Otherwise, there exists a line segment in admissible cone apexed either at a vertex of $P_i$ or at a vertex of $P_j$ which yield a shorter path from source $s$ to $q$ without using $pq$ line segment. ∎

Once we find a shortest path $SP_\Omega$ amid obstacles in $\Omega$ using spanner $G$, as for the proof of Lemma 2.3, we transform it to a shortest path amid obstacles in $P$. Since there are $O(h)$ obstacles in $\Omega$, $SP_\Omega$ contains $O(h)$ tangents. Let this set of tangents be $T$. We need to find points of intersection of convex polygons in $P$ with line segments in $T$. Whenever a line segment $l \in T$ and a convex polygon $P_i \in P$ 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 $l \in SP_\Omega - T$ 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 [8] to find the points of intersections of line segments in $T$ with the convex obstacles in $P$.

As part of 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 $P$. Initially, points in $L$ and $R$ together with the two endpoints of every line segment in $T$ are inserted into the priority queue, say $Q$. The event points are scheduled from $Q$ using their distance from where the sweep line is initially placed as priority. As the algorithm progress, the event points corresponding to $L, R$, and the endpoints of line segments in $T$ are deleted from $Q$. The algorithm terminates whenever the $Q$ is empty. As described below, the intersection points between the line segments in $T$ and the convex polygons in $P$ are added to $Q$ with the traversal of sweep line. The sweep line status is maintained as a balanced binary search tree $B$. We insert (resp. delete) a line segment in $T$ or convex polygon in $P$, say $s$, to $B$ whenever leftmost (resp. rightmost) endpoint of $s$ is popped from $Q$. Since before an intersection occurs between a line segment $l$ from $T$ and a convex polygon $P$ from $P$, it is guaranteed that $l$ and $P$ occur adjacent along the sweep line, we update event-point schedule with an intersection between $l$ and $P$ whenever $l$ and $P$ are adjacent in the sweep line status. By using the algorithm from Dobkin et al. [9], 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 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
interjection point. Further, whenever \( l \) and \( P \) become non-adjacent along the sweep line, we delete their leftmost point of intersection from \( Q \).

**Theorem 2.1** Given \( P \) and two points \( s \) and \( t \) in \( F(P) \), computing a \((1+\epsilon)\)-approximate distance between \( s \) and \( t \) takes \( O(n + \frac{h}{\sqrt{\epsilon'}} \log \frac{h}{\epsilon'}) \) time for \( \epsilon' = \sqrt{1+\epsilon} - 1 \). Further, within an additional \( O(h \log n) \) time, a \((1+\epsilon)\)-approximate shortest path is computed.

**Proof:** From Lemma 2.4, we know that \( dist_P(s,t) \leq dist_{\Omega}(s,t) \leq (1+\alpha)dist_P(s,t) \). Let \( G \) be the spanner constructed. From Lemma 2.6, we know that \( dist_{\Omega}(s,t) \leq dist_G(s,t) \leq (1+\alpha)dist_{\Omega}(s,t) \). Combining these two inequalities yields \( dist_P(s,t) \leq dist_G(s,t) \leq (1+\alpha)^2 dist_P(s,t) \). To achieve \((1+\epsilon)\)-approximation, we set \( \alpha = \frac{1}{\sqrt{1+\epsilon}} - 1 \). From here on, we denote \( \alpha \epsilon \) with \( \epsilon' \).

Finding coreset vertices and computing corepolygons takes \( O(n) \) time. The number of coreset vertices is \( O\left(\frac{h}{\sqrt{\epsilon'}}\right) \). This 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 \( O\left(\frac{h}{\sqrt{\epsilon'}}\right) \). Finding a shortest path between \( s \) and \( t \) in \( G \) takes \( O\left(\frac{h}{\sqrt{\epsilon'}} \log \frac{h}{\sqrt{\epsilon'}}\right) \) time. Hence, computing closest Steiner points corresponding to all the cone orientations in \( \Omega \) together take \( O\left(\frac{h}{\sqrt{\epsilon'}} \log \frac{h}{\sqrt{\epsilon'}}\right) \).

The number of nodes in the spanner \( G \) is \( O\left(\frac{h}{\sqrt{\epsilon'}}\right) \). This 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 \( O\left(\frac{h}{\sqrt{\epsilon'}}\right) \). Finding a shortest path between \( s \) and \( t \) in \( G \) takes \( O\left(\frac{h}{\sqrt{\epsilon'}} \log \frac{h}{\sqrt{\epsilon'}}\right) \) time. Hence, computing the \((1+\epsilon')\)-approximate distance between \( s \) and \( t \) takes \( O(n + \frac{h}{\sqrt{\epsilon'}} \log \frac{h}{\sqrt{\epsilon'}}) \) time.

For the plane sweep, leftmost and rightmost extreme vertices of convex polygons in \( P \) are found in \( O(n) \) time. There are \( O(h) \) line segments in \( T \), cardinality of \( \Omega \) is \( O(h) \), and \( O(h) \) line segment-obstacle pairs (respectively from \( T \) and \( P \)) that intersect. The number of event points due to \( L, R \), and endpoints of line segments in \( T \) 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 structure is updated at the points in \( R \) as well as with the rightmost endpoints of line segments in \( T \); there are \( O(h) \) such event points. Analogous to the analysis provided for line segment intersection [3], our plane sweep algorithm takes \( O(n + h \log h) \) time.

Due to Dobkin et al. [9], determining whether a line segment \( l \) in \( SP_{\Omega} \) intersects with an obstacle \( P \) takes \( O(\log n) \) time. The preprocessing structures corresponding to [9] take \( O(n) \) space and they are constructed in \( O(n) \) time. Further, replacing every line segment between points of intersection with the corresponding geodesic shortest path along the boundaries of obstacles together takes \( O(n) \) time altogether. \( \square \)

### 3 Computing approximate shortest path in polygonal domain

In this section, we extend the approximation method to the case of non-convex polygons. We reduce the problem of computing approximate Euclidean shortest path amid simple polygonal obstacles to that of computing the approximate Euclidean shortest path amid convex polygonal obstacles. This is accomplished by first decomposing \( F(P) \) into a set of corridors, funnels, hourglasses, and junctions [14][18][16]. First, we describe these geometric structures and then detail how these help in the reduction.
For convenience, obstacles of $\mathcal{P}$ are assumed to be contained in a rectangle $\mathcal{R}$. Let $\text{Tri}(\mathcal{F})$ denote a triangulation of $\mathcal{F}(\mathcal{P})$. (Refer Fig. 5) The line segments of $\text{Tri}(\mathcal{F})$ that are not obstacle edges are referred to as diagonals. Let $G(\mathcal{F})$ denote the dual graph of $\text{Tri}(\mathcal{F})$, i.e., each node of $G(\mathcal{F})$ corresponds to a triangle of $\text{Tri}(\mathcal{F})$ and each edge connects two nodes corresponding to two triangles sharing a diagonal of $\text{Tri}(\mathcal{F})$. 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 resulting graph is $G^3$ (refer Fig. 5), which has $O(h)$ faces, nodes, and edges. Every node of $G^3$ corresponds to a triangle in $\text{Tri}(\mathcal{F})$, called a junction triangle. (Refer Fig. 5) The removal of the nodes for all junction triangles from $G^3$ results in $O(h)$ corridors, each of which corresponds to an edge of $G^3$. This procedure leaves points $s$ and $t$ between which geodesic Euclidean shortest path needs to be computed are to be in their own 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 $\pi(a, b)$ (resp., $\pi(e, f)$) be the Euclidean shortest path from $a$ to $b$ (resp., $e$ to $f$) in $C$. The region $H_C$ bounded by $\pi(a, b), \pi(e, f), \overline{fe}$, and $\overline{fa}$ is called an hourglass, which is open if $\pi(a, b) \cap \pi(e, f) = \emptyset$ and closed otherwise. (Refer Fig. 6) If $H_C$ is open, then both $\pi(a, b)$ and $\pi(e, f)$ are convex polygonal chains and are called the sides of $H_C$; otherwise, $H_C$ consists of two funnels and a path $\pi_C = \pi(a, b) \cap \pi(e, f)$ joining the two apices of the two funnels, and $\pi_C$ is called the corridor path of $C$. The paths $\pi(b, x), \pi(e, x), \pi(a, y)$, and $\pi(f, y)$ are termed sides of funnels of hourglass $H_C$. The sides of funnels are convex polygonal chains. For any obstacle $P_j \in \mathcal{P}$, let $\pi(b, x)$ be a side $S$ of a funnel $F$ of $H_C$ such that $b$ is a vertex of $P_j$. Let $\pi(b, x')$ be a maximal subpath of $\pi(b, x)$ such that $x'$ is a vertex of $P_j$. Then $x'$ is termed a pseudo-apex of side $S$ of funnel $F$. Note that pseudo-apex of a side $S$ could be same as the apex of the funnel $F$. Let $x', x''$ be two pseudo-apices of a closed hourglass $H_C$ such that $x'$ and $x''$ are vertices of $P_j$. The shortest path between $x'$ and $x''$ along the boundary of $H_C$ is the corridor path between pseudo-apices.

![Figure 5](image1.png)  
**Figure 5:** Illustrating a triangulation of the free space among two obstacles and the corridors (indicated by red solid curves). There are two junction triangles marked by a large dot inside each of them, connected by three solid (red) curves. Removing the two junction triangles results in three corridors.

![Figure 6](image2.png)  
**Figure 6:** Illustrating an open hourglass (left) and a closed hourglass (right) with a corridor path connecting the apices $x$ and $y$ of the two funnels. The dashed segments are diagonals.
closed and bounded polygon defined by polygonal chains in $C_j$ contains $P_j$; hence, we term this new polygon as the expanded polygon of $P_j$. If every chain in $C_j$ is convex, then $\bigcup_{C \in C_j} C$ is $CH(P_j)$. Since $s$ and $t$ lie exterior to closed hourglasses, as described below, in forming the spanner graph $G$, we handle corridor paths between pseudo-apices as a special case and introduce corresponding edges into $G$.

For every convex polygonal chain $C$ of every open hourglass that correspond to any polygonal obstacle $P_j$, endpoints of $C$ are chosen to be in the coreset $S_j$ corresponding to $P_j$. Further, similar to the case of convex polygonal obstacles, we compute a coreset of every such convex polygonal chain $C$ and replace $C$ with the convex polygonal chain $C'$ that passes through those coreset of vertices. Essentially, we partition $C$ into patches with angle subtended by each patch to be same as in the case of convex polygonal obstacles. (For details, refer Section 2.) For every closed hourglass $H_C$ that has a vertex common to $P_j$, both the pseudo-apices of $H_C$ that incident to $P_j$ are included into coreset of $P_j$. Similar to convex polygonal chains of open hourglasses, coresets corresponding to maximal sections of sides of funnels whose endpoints incident to $P_j$ are computed; and, convex polygonal chain that passes through these coreset of vertices of $C$ is used. Excluding line segments joining successive pseudo-apices, the convex polygonal chains that pass through coreset vertices of each polygonal chain of $P_j$ bound a closed region termed corepolygon $P_j^\prime$ of $P_j$. Since there are $O(h)$ open and closed corridors together and since there are $O(1)$ convex sides per corridor the following is immediate.

**Observation 2** The size of coreset $S$ of $P$ is $O(\frac{h}{\sqrt{\text{in} t})}$.

Similar to convex polygonal obstacle case, we construct a spanner $G(V, E)$ that correspond to corepolygons of $P$ using CVDs. It is immediate to note that Clarkson’s method extends to corepolygons defined as above. And, for each line segment that joins two pseudo-apices along a corepolygon, an edge $e(v', v'')$ is introduced into $G$: here, $v'$ and $v''$ are the nodes corresponding to endpoints of $C$; and, the weight of $e$ is set as the distance between $v'$ and $v''$ along $C$, which is the length of $C$. Since both the pseudo-apices are included in the coreset of the corresponding obstacle, $v', v'' \in V$. For a shortest path $p$ between any two nodes of $G$, before outputting $p$, for every edge $e \in p$ if both the endpoints of $e$ correspond to pseudo-apices $a', a''$ of a closed hourglass then we replace $p$ with the shortest path along the corridor path between $a'$ and $a''$. Computing hourglasses of $F(P)$ using [14, 15, 16] takes $O(n + h(\lg h)^{1+\delta} + h \lg n)$ time (where $\delta$ is a small positive constant due to time involved in triangulating $F(P)$ using [15]). Extending the proof of Theorem 2.1 leads to the following.

**Theorem 3.1** Given a set $P$ of polygonal obstacles and two points $s$ and $t$ in $F(P)$, computing a $(1+ \epsilon)$-approximate Euclidean shortest path between $s$ and $t$ in $O(n + h((\lg n) + (\lg h)^{1+\delta} + (\frac{1}{\sqrt{\epsilon}} \lg \frac{h}{\delta})))$ time. Here $\epsilon'$ is $\sqrt{1+\epsilon}$, and $\delta$ is a small positive constant.

## 4 Two-point approximate distance queries amid convex polygons

Our query algorithm constructs an auxiliary graph from the spanner network computed during the preprocessing phase of the algorithm. Like in the previous section, our preprocessing algorithm relies on [3]. 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 a geodesic shortest path between any two vertices in $\mathcal{P}$. However, since the time involved in finding a shortest path between nodes in $G$, or the space to save all-pair-shortest-paths in $G$ are resource intensive, we compute a planar graph $G^{pl}(V, E^{pl})$ from $G(V, E)$ using the result from Chew [4]. 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. [19] to efficiently answer $(1+\epsilon)$-approximate distance (length) queries in $G^{pl}$. More specifically, [19] takes $O(|V| \lg^2 |V|)$ time to construct a data structure of size $O(|V|)$ so that $(2+\epsilon)$-approximate shortest distance queries are answered in $O(\frac{|V|^2}{\epsilon^2})$ time.

**Lemma 4.1** Let $G$ be the spanner computed for the polygonal domain $\Omega$ using the algorithm mentioned in Subsection 2.2. Let $s, t$ be two points in $\mathcal{P}$. Let $G^{pl}$ be the planar graph constructed from $G$ using [4]. Further, let $\text{dist}_K(s, t)$ be the distance between $s$ and $t$ in $G^{pl}$ computed using the algorithm from [19]. With appropriately chosen parameters, $\text{dist}_\mathcal{P}(s, t) \leq \text{dist}_K(s, t) \leq (2+\epsilon)\text{dist}_\mathcal{P}(s, t)$.

**Proof:** From Lemma 2.4 we know that $\text{dist}_\mathcal{P}(s, t) \leq \text{dist}_\Omega(s, t) \leq (1+\alpha)\text{dist}_\mathcal{P}(s, t)$. Let $\text{dist}_G(s, t)$ be the distance in $G$ between nodes $s$ and $t$ of $G$. From Lemma 2.6 we know that $\text{dist}_\Omega(s, t) \leq \text{dist}_G(s, t) \leq (1+\alpha)\text{dist}_\Omega(s, t)$. Let $\text{dist}_{G^{pl}}(s, t)$ be the distance in $G^{pl}$ between nodes $s$ and $t$ of $G^{pl}$. From [4], $\text{dist}_{G^{pl}}(s, t) \leq \text{dist}_G(\psi(s), \psi(t)) \leq 2\text{dist}_G(s, t)$. Then, as mentioned above, $\text{dist}_{G^{pl}}(s, t) \leq \text{dist}_K(s, t) \leq (1+\alpha)\text{dist}_{G^{pl}}(s, t)$. Combining these, $\text{dist}_\mathcal{P}(s, t) \leq \text{dist}_K(s, t) \leq (1+\alpha)^3(2)\text{dist}_\mathcal{P}(s, t)$. To achieve $(2+\epsilon)$-approximation, we set $\alpha$ to $\frac{1}{6}(\frac{2+\epsilon}{2})^{1/3} - 1$. Since $\epsilon < -1$ for $\epsilon < 1$, algorithm yields a $(2+\epsilon)$-approximation. □

In the following, we denote $\alpha \epsilon = (\frac{2+\epsilon}{2})^{1/3} - 1$ with $\epsilon''$. From here on, we suppose that there are $O(\frac{1}{\sqrt{\epsilon''}})$ cones in $C$, each cone with a cone angle $O(\sqrt{\epsilon''})$. It remains to describe data structures that need to be constructed during the preprocessing phase so as to obtain the closest vertex of query point $s$ (resp. $t$) in a given cone $C_s$ (resp. $C_t$). The significance of the same is explained later. To efficiently determine all these $O(\frac{1}{\sqrt{\epsilon''}})$ neighbors to $s$ and $t$ during query time, we construct a set of $O(\frac{1}{\sqrt{\epsilon''}})$ cone Voronoi diagrams (CVDs), each of which correspond to a cone in $C$. For any cone $C \in \mathcal{C}$ with orientation $\psi$, the CVD in that orientation is denoted with $CVD_\psi$. For each $C \in \mathcal{C}$, we consider a sweep-line orthogonal to the orientation of $C$. The sweep-line algorithm details are same as mentioned in Subsection 2.2.

**Lemma 4.2** The preprocessing phase takes $O(n + \frac{h}{\epsilon''} \lg \frac{h}{\sqrt{\epsilon''}} + \frac{h}{\sqrt{\epsilon''}} \lg^2 \frac{h}{\sqrt{\epsilon''}})$ time. The space complexity of the data structures constructed by the end of preprocessing phase is $O(\frac{h}{\sqrt{\epsilon''}})$. Here, $\epsilon''$ is $(\frac{2+\epsilon}{2})^{1/3} - 1$.

**Proof:** Computing sketch $\Omega$ of $\mathcal{P}$ takes $O(n + \frac{h}{\sqrt{\epsilon''}})$ time. The number of cones in all CVDs is $O(\frac{h}{\sqrt{\epsilon''}})$. Constructing spanner $G$ involve computing CVDs, together it takes $O(\frac{1}{\sqrt{\epsilon''}} \frac{h}{\sqrt{\epsilon''}} \lg \frac{h}{\sqrt{\epsilon''}})$ time to compute $G$. Due to [4], computing planar graph $G^{pl}$ with $O(\frac{h}{\sqrt{\epsilon''}})$ nodes takes $O(\frac{h}{\sqrt{\epsilon''}} \lg \frac{h}{\sqrt{\epsilon''}})$ time. Computing space-efficient data structures using [19] take $O(\frac{h}{\sqrt{\epsilon''}} \lg^2 \frac{h}{\sqrt{\epsilon''}})$ time. Hence, the preprocessing phase takes $O(n + \frac{h}{\sqrt{\epsilon''}} \lg \frac{h}{\sqrt{\epsilon''}} + \frac{h}{\sqrt{\epsilon''}} \lg^2 \frac{h}{\sqrt{\epsilon''}})$ time.

Further, data structures constructed using [19] by the end of preprocessing phase occupy $O(\frac{h}{\sqrt{\epsilon''}})$ space. Using Kirkpatrick’s point location [20], data structures for planar point location are of
The time to query the path between Lemma 4.3
The shortest path obtained between $s$ and $t$ in $P$ available before preprocessing the $(2+\epsilon)$ maintained due to [19].
Computing shortest path between $s$ and $t$ to some vertex in coreset $t$, the same is true with the edges of type $(\{s\} \times V_t)$ with $s' \in V_s$, the weight of edge $(s, s')$ is as defined above. The same is true with the edges of type $(t, t')$ with $t' \in V_t$. Note that every node $v \in V_s \cup V_t$ correspond to some vertex in coreset $S$. 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 data structures maintained due to [19].

We apply Fredman-Tarjan algorithm to find a shortest path between $s$ and $t$ in $G_{st}$. It follows from the above discussion, that this distance is a $(2+\epsilon)$-approximate distance from $s$ to $t$ amid $P$. Computing shortest path between $s$ and $t$ in this way effectively equivalent to having both $s$ and $t$ available before preprocessing $P$. Therefore, the following is immediate from Lemma 4.1.

**Lemma 4.3** The shortest path obtained between $s$ and $t$ in $G_{st}$ is a $(2 + \epsilon)$-approximate shortest path between $s$ and $t$ amid set $P$ of convex polygons.

**Lemma 4.4** The time to query the $(2 + \epsilon)$-approximate distance between two vertices in $P$ is $O\left(\frac{1}{\epsilon'^2} \log^2 \frac{h}{\sqrt{\epsilon''}}\right)$.

**Proof:** Locating a query point $s$ (resp. $t$) in any one $CVD_\psi$ takes $O(\log \frac{h}{\sqrt{\epsilon''}})$ time. Since there are $O\left(\frac{1}{\epsilon''} \right)$ CVDis and since finding a closest Steiner point (or, vertex) in each such CVD takes $O(\log \frac{h}{\sqrt{\epsilon''}})$ time, computing all the closest Steiner points (or, vertices) of $s$ and $t$ together take $O\left(\frac{1}{\epsilon''} \log \frac{h}{\sqrt{\epsilon''}}\right)$. The number of nodes and the number of edges of $G_{st}$ are respectively $O\left(\frac{1}{\epsilon''} \right)$ and $O\left(\frac{1}{\epsilon''} \right)$. Using [19], finding the distance between any two nodes in $G^{pl}$ take $O\left(\frac{1}{\epsilon'^2} \log^2 \frac{h}{\sqrt{\epsilon''}}\right)$ time. There are $O\left(\frac{1}{\epsilon''} \right)$ pairs of neighbors of $s$ and $t$ that needs to be considered in computing approximate distances in $G^{pl}$. Fredman-Tarjan algorithm to compute a shortest path in $G_{st}$ from $s$ to $t$ takes $O\left(\frac{1}{\epsilon''} \log^* \frac{1}{\epsilon''}\right)$. The Theorem follows when these asymptotic time complexities are added up. \qed

**Theorem 4.1** Given a set $P$ of $h$ pairwise-disjoint convex polygons of total complexity $n$ in $\mathbb{R}^2$, preprocess $P$ in $O(n + \frac{h}{\epsilon''} \log \frac{h}{\sqrt{\epsilon''}} + \frac{h}{\epsilon''} \log^2 \frac{h}{\sqrt{\epsilon''}})$ time to build a data structure of size $O\left(\frac{h}{\sqrt{\epsilon''}}\right)$. For any two query points $s, t \in F(P)$, a $(2 + \epsilon)$-approximate distance is found in $O\left(\frac{1}{\epsilon'^2} \log^2 \frac{h}{\sqrt{\epsilon''}}\right)$ time. Here, $\epsilon'' = (\frac{1 + \epsilon}{2})^{1/3} - 1$. 

13
5 Conclusions

We have presented an algorithm to compute a $(1 + \epsilon)$-approximate Euclidean shortest path amid polygonal obstacles in $\mathbb{R}^2$. The time and space complexities of our algorithm better the algorithms devised in Chen [3]. We also devised a $(2 + \epsilon)$-approximation algorithm to answer two-point Euclidean distance queries. Our result trades off with approximation algorithms devised in [2, 3]. These algorithms rely on computing a sketch of the polygonal domain by extracting coresets of polygonal obstacles and further computing corepolygons from each of these coresets.

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] D. Z. Chen. On the All-Pairs Euclidean Short Path Problem. In *Proceedings of Symposium on Discrete Algorithms*, pages 292–301, 1995.

[4] 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.

[5] 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.

[6] K. L. Clarkson. Approximation Algorithms for Shortest Path Motion Planning (Extended Abstract). In *Proceedings of Symposium on Theory of Computing*, pages 56–65, 1987.

[7] H. Guo and A. Maheshwari and J-R. Sack Shortest Path Queries in Polygonal Domains. In *Proceedings of Algorithm Aspects in Information and Management*, pages 200–211, 2008.

[8] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag TELOS, Santa Clara, CA, USA, 3rd ed. edition, 2008.

[9] D. P. Dobkin and David G. Kirkpatrick. Fast Detection of Polyhedral Intersection. *Theoretical Computer Science*, 27:241–253, 1983.

[10] S. K. Ghosh. *Visibility Algorithms in the Plane*. Cambridge University Press, New York, NY, USA, 2007.

[11] 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.

[12] J. Hershberger and S. Suri. An Optimal Algorithm for Euclidean Shortest Paths in the Plane. *SIAM Journal on Computing*, 28(6):2215–2256, 1999.

[13] 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.
[14] S. Kapoor, and S. N. Maheshwari. Efficient algorithms for Euclidean shortest path and visibility problems with polygonal obstacles. In Proceedings of Symposium on Computational Geometry, pages 172–182, 1988.

[15] B. Chazelle. Triangulating a simple polygon in linear time. Discrete & Computational Geometry, 6(5):485–524, 1991.

[16] S. Kapoor. Efficient Computation of Geodesic Shortest Paths. In Proceedings of Symposium on Theory of Computing, pages 770–779, 1999.

[17] S. Kapoor and S. N. Maheshwari. Efficiently Constructing the Visibility Graph of a Simple Polygon with Obstacles. SIAM Journal on Computing, 30(3):847–871, 2000.

[18] 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.

[19] K. Kawarabayashi, P. N. Klein, and C. Sommer. Linear-Space Approximate Distance Oracles for Planar, Bounded-Genus and Minor-Free Graphs. In Proceedings of International Colloquium on Automata, Languages and Programming, pages 135–146, 2011.

[20] D. G. Kirkpatrick. Optimal Search in Planar Subdivisions. SIAM Journal on Computing, 12(1):28–35, 1983.

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

[22] J. S. B. Mitchell. Geometric Shortest Paths and Network Optimization. In Handbook of Computational Geometry, pages 633–701. Elsevier Science Publishers B.V. North-Holland, 1998.

[23] E. Welzl. Constructing the Visibility Graph for n-Line Segments in O(n^2) Time. Information Processing Letters, 20(4):167–171, 1985.

[24] A. C. Yao. On Constructing Minimum Spanning Trees in k-Dimensional Spaces and Related Problems. SIAM Journal on Computing, 11(4):721–736, 1982.