TWO-POINT L_1 SHORTEST PATH QUERIES IN THE PLANE*

Danny Z. Chen,[†] *Rajasekhar Inkulu,*[‡] *and Haitao Wang*[§]

Abstract. Let $\mathcal{P}$ be a set of h pairwise-disjoint polygonal obstacles with a total of n vertices in the plane. We consider the problem of building a data structure that can quickly compute an L_1 shortest obstacle-avoiding path between any two query points s and t. Previously, a data structure of size $O(n^2 \log n)$ was constructed in $O(n^2 \log^2 n)$ time that answers each two-point query in $O(\log^2 n + k)$ time, i.e., the shortest path length is reported in $O(\log^2 n)$ time and an actual path is reported in additional $O(k)$ time, where k is the number of edges of the output path. In this paper, we build a new data structure of size $O(n+h^2 \cdot \log h \cdot 4^{\sqrt{\log h}})$ in $O(n+h^2 \cdot \log^2 h \cdot 4^{\sqrt{\log h}})$ time that answers each query in $O(\log n + k)$ time. (In contrast, for the Euclidean version of this two-point query problem, the best known algorithm uses $O(n^{11})$ space to achieve an $O(\log n + k)$ query time.) Further, we extend our techniques to the weighted rectilinear version in which the "obstacles" of $\mathcal{P}$ are rectilinear regions with "weights" and allow L_1 paths to travel through them with weighted costs. Previously, a data structure of size $O(n^2 \log^2 n)$ was built in $O(n^2 \log^2 n)$ time that answers each query in $O(\log^2 n + k)$ time. Our new algorithm answers each query in $O(\log n + k)$ time with a data structure of size $O(n^2 \cdot \log n \cdot 4^{\sqrt{\log n}})$ that is built in $O(n^2 \cdot \log^2 n \cdot 4^{\sqrt{\log n}})$ time.

1 Introduction

Let $\mathcal{P}$ be a set of h pairwise-disjoint polygonal obstacles in the plane with a total of n vertices. We consider two-point shortest obstacle-avoiding path queries for which the path lengths are measured in the L_1 metric. The plane minus the interior of the obstacles is called the *free space*. Our goal is to build a data structure to quickly compute an L_1 shortest path in the free space between any two query points s and t. Previously, Chen *et al.* [7] constructed a data structure of size $O(n^2 \log n)$ in $O(n^2 \log^2 n)$ time that computes the length of the L_1 shortest s-t path in $O(\log^2 n)$ time and an actual path in additional $O(k)$ time, where k is the number of edges of the output path.

Throughout this paper, unless otherwise stated, when we say that the query time of a data structure is $O(f(n, h))$ (which may be a function of both n and h), we mean that the

*A preliminary version appeared in the Proceedings of the 30th Annual Symposium on Computational Geometry (SoCG 2014).

†*Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, USA*, dchen@nd.edu

‡*Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, Guwahati 781039, Assam, India*, rinkulu@iitg.ac.in

§Corresponding author. *Department of Computer Science, Utah State University, Logan, UT 84322, USA*, haitao.wang@usu.edu

shortest path length can be reported in $O(f(n,h))$ time and an actual path can be found in additional time linear in the number of edges of the output path.

In this paper, we build a new data structure of size $O(n + h^2 \cdot \log h \cdot 4^{\sqrt{\log h}})$ in $O(n + h^2 \cdot \log^2 h \cdot 4^{\sqrt{\log h}})$ time, with $O(\log n)$ query time. Note that $n + h^2 \cdot \log^2 h \cdot 4^{\sqrt{\log h}} = O(n + h^{2+\epsilon})$ for any constant $\epsilon > 0$. Comparing with the results in [7], we reduce the query time by a logarithmic factor, and use less preprocessing time and space when h is small, e.g., $h = O(n^\delta)$ for any constant $\delta < 1$. In particular, if $h = O(n^\delta)$ for any constant $\delta < 1/2$, then the size and the construction time of our data structure are both bounded by $O(n)$.

Further, we extend our techniques to the *weighted rectilinear version* in which each "obstacle" $P \in \mathcal{P}$ is a region with a nonnegative weight $w(P)$ and the edges of the obstacles in $\mathcal{P}$ are all axis-parallel; a path intersecting the interior of P is charged a cost depending on $w(P)$. For this problem, Chen *et al.* [7] constructed a data structure of size $O(n^2 \log^2 n)$ in $O(n^2 \log^2 n)$ time that answers each two-point shortest path query in $O(\log^2 n)$ time. We build a new data structure of size $O(n^2 \cdot \log n \cdot 4^{\sqrt{\log n}})$ in $O(n^2 \cdot \log^2 n \cdot 4^{\sqrt{\log n}})$ time that answers each query in $O(\log n)$ time. Note that $n^2 \cdot \log^2 n \cdot 4^{\sqrt{\log n}} = O(n^{2+\epsilon})$ for any $\epsilon > 0$.

1.1 Related Work

The problems of computing shortest paths among obstacles in the plane have been studied extensively (e.g., [6, 7, 8, 9, 12, 11, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 31, 34, 35, 36, 37, 38, 39]). There are three main types of problems: *finding a single shortest s-t path* (both s and t are given as part of the input and the goal is to find a single shortest s-t path), *single-source shortest path queries* (s is given as part of the input and the goal is to build a data structure to answer shortest path queries for any query point t), and *two-point shortest path queries* (as defined and considered in this paper). The distance metrics can be the Euclidean (i.e., L_2) or L_1. Refer to Mitchell [40] for a comprehensive survey on this topic.

For the simple polygon case, in which $\mathcal{P}$ is a single simple polygon, all three types of problems have been solved optimally [20, 21, 22, 24, 34], in both the Euclidean and L_1 metrics. Specifically, an $O(n)$-size data structure can be built in $O(n)$ time that answers each two-point Euclidean shortest path query in $O(\log n)$ time [20, 22]. Since in a simple polygon a Euclidean shortest path is also an L_1 shortest path [24], the results in [20, 22] hold for the L_1 metric as well.

The polygonal domain case (or "a polygon with holes"), in which $\mathcal{P}$ has h obstacles as defined above, is more difficult. For the Euclidean metric, Hershberger and Suri [25] built a *single source shortest path map* of size $O(n \log n)$ in $O(n \log n)$ time that answers each query in $O(\log n)$ time. For the L_1 metric, Mitchell [36, 38] built an $O(n)$-size single source shortest path map in $O(n \log n)$ time that answers each query in $O(\log n)$ time. Later, Chen and Wang [8, 9, 11] built an L_1 single source shortest path map of size $O(n)$ in $O(n+h \log h)$ time, with an $O(\log n)$ query time, for a triangulated free space (the current best triangulation algorithm runs in $O(n \log n)$ or $O(n + h \log^{1+\epsilon} h)$ time for any constant $\epsilon > 0$ [2]). For two-point L_1 shortest path queries, Chen *et al.* [7] gave the previously best solution, as mentioned above; for a special case where the obstacles are rectangles, ElGindy and Mitra [19] gave an $O(n^2)$ size data structure that supports $O(\log n)$ time queries. For two-point queries in the Euclidean metric, Chiang and Mitchell [15] constructed a data structure of size $O(n^{11})$ that answers each query in $O(\log n)$ time, and alternatively, a data structure of size $O(n + h^5)$ with an $O(h \log n)$ query time; other data structures with trade-off between preprocessing and query time were also given in [15]. If the query points s and t are both restricted to the boundaries of the obstacles of $\mathcal{P}$, Bae and Okamoto [1] built a data structure of size $O(n^5 \text{poly}(\log n))$ that answers each query in $O(\log n)$ time, where $\text{poly}(\log n)$ is a polylogarithmic factor. Efficient algorithms were also given for the case when the obstacles have curved boundaries [6, 12, 14, 23, 26].

For the weighted region case, in which the "obstacles" allow paths to pass through their interior with weighted costs, Mitchell and Papadimitriou [41] gave an algorithm that finds a weighted Euclidean shortest path in a time of $O(n^8)$ times a factor related to the precision of the problem instance. Carufel et al. [3] shows that the problem essentially cannot be solved in the algebraic computation model over the rational numbers. For the weighted rectilinear case, Lee *et al.* [35] presented two algorithms for finding a weighted L_1 shortest path, and Chen *et al.* [7] gave an improved algorithm with $O(n \log^{3/2} n)$ time and $O(n \log n)$ space. Chen *et al.* [7] also presented a data structure for two-point weighted L_1 shortest path queries among weighted rectilinear obstacles, whose performance has already been discussed above.

1.2 Our Approaches

Our first main idea is to propose an enhanced graph model based on the scheme in [7, 16, 17], to reduce the query time from $O(\log^2 n)$ to $O(\log n)$. In [7, 16, 17], to build a graph, a total of n vertical lines (called "cut-lines") are created recursively in $O(\log n)$ levels. Then, each obstacle vertex v is projected to $O(\log n)$ cut-lines (one cut-line per level) to create "Steiner points" if v is horizontally visible to such cut-lines. For any two query points s and t, to report an L_1 shortest s-t path, the algorithm in [7] finds $O(\log n)$ Steiner points (called "gateways") on $O(\log n)$ cut-lines for each of s and t, such that there must be a shortest s-t path containing a gateway of s and a gateway of t. Consequently, a shortest path is obtained in $O(\log^2 n)$ time using the $O(\log n)$ gateways of s and t.

We propose an enhanced graph G_E by adding more Steiner points onto the cut-lines such that we need only $O(\sqrt{\log n})$ gateways for any query point, and consequently, computing the shortest path length takes $O(\log n)$ time. More specifically, for each obstacle vertex, instead of projecting it to a single vertical cut-line at each level, we project it to $O(2^{\sqrt{\log n}})$ cut-lines in every $O(\sqrt{\log n})$ consecutive levels (thus creating $O(2^{\sqrt{\log n}})$ Steiner points); in fact, these cut-lines form a binary tree structure of height $O(\sqrt{\log n})$ and they are carefully chosen to ensure that $O(\sqrt{\log n})$ gateways are sufficient for any query point. Hence, the size of the graph G_E is $O(n\sqrt{\log n}2^{\sqrt{\log n}})$.

To improve the data structure construction so that its time and space bounds depend linearly on n, we utilize the extended corridor structure [8, 9, 11], which partitions the free space of $\mathcal{P}$ into an "ocean" $\mathcal{M}$, and multiple "bays" and "canals". We build a graph $G_E(\mathcal{M})$ of size $O(h\sqrt{\log h}2^{\sqrt{\log h}})$ on $\mathcal{M}$ similar to G_E, such that if both query points are in $\mathcal{M}$, then the query can be answered in $O(\log n)$ time. It remains to deal with the general case when at least one query point is not in $\mathcal{M}$. This is a major difficulty of our approach and our algorithm for this case is another of our main contributions. Below, we use a bay as an example to illustrate our main idea for this algorithm.

For two query points s and t, suppose s is in a bay B and t is outside B. Since B is a simple polygon, any shortest s-t path must cross the "gate" g of B, which is a single edge shared by B and $\mathcal{M}$. We prove that there exists a shortest s-t path that must contain one of three special points $z(s)$, $z_1(s)$, and $z_2(s)$, where $z(s)$ is in B and the other two points are on g (and thus in $\mathcal{M}$). For the case when a shortest s-t path contains either $z_1(s)$ or $z_2(s)$, we can use the graph $G_E(\mathcal{M})$ to find such a shortest path. For the other case, we build another graph $G_E(g)$ based on the horizontal projections of the vertices of $G_E(\mathcal{M})$ on g, and use $G_E(g)$ to find such a shortest path (along with a set of interesting observations) by a merge of $G_E(g)$ and $G_E(\mathcal{M})$. Intuitively, $G_E(g)$ plays the role of connecting the shortest path structure inside B with those in $\mathcal{M}$.

The case when a query point is in a canal can be handled similarly, although it is more complicated because each canal has two gates.

The rest of the paper is organized as follows. In Section 2, we introduce notation and sketch the previous results that will be needed by our algorithms. In Section 3, we propose our enhanced graph G_E that helps reduce the query time to $O(\log n)$. In Section 4, we further reduce the preprocessing time and space by using the extended corridor structure. In Section 5, we extend our techniques in Section 3 to the weighted rectilinear case.

Henceforth, unless otherwise stated, "shortest paths" always refer to L_1 shortest paths and "distances" and "lengths" always refer to L_1 distances and lengths. To distinguish from graphs, the vertices/edges of $\mathcal{P}$ are always referred to as obstacle vertices/edges, and graph vertices are referred to as "nodes". For simplicity of discussion, we make a general position assumption that no two obstacle vertices have the same x- or y-coordinate except for the weighted rectilinear case. This assumption is made without loss of generality, since we can always perturb the input slightly to achieve the assumption, as does in [38].

2 Preliminaries

A path in the plane is *x-monotone* (resp., *y-monotone*) if its intersection with any vertical (resp., horizontal) line is either empty or connected. A path is *xy-monotone* if it is both *x-monotone* and *y-monotone*. Note that any xy-monotone path is an L_1 shortest path.

A point p is *visible* to another point q if the line segment $\overline{pq}$ is in the free space. A point p is *horizontally visible* to a line l if there is a point q on l such that $\overline{pq}$ is horizontal and is in the free space. For a line l and a point p, the point $q \in l$ is the *horizontal projection* of p on l if $\overline{pq}$ is horizontal, and we denote it by $p_h(l) = q$. Let $\partial\mathcal{P}$ denote the boundaries of all obstacles in $\mathcal{P}$. For a point p in the free space of $\mathcal{P}$, if we shoot a horizontal ray from p to the left, the first point on $\partial\mathcal{P}$ hit by the ray is called the *leftward projection* of p on $\partial\mathcal{P}$, denoted by p^l; similarly, we define the *rightward, upward*, and *downward* projections of p on $\partial\mathcal{P}$, denoted by p^r, p^u, and p^d, respectively.

We sketch the graph in [7], denoted by G_{old}, for answering two-point queries, which was originally proposed in [16, 17] for computing a single shortest path. To define G_{old}, two types of *Steiner points* are specified, as follows. For each obstacle vertex p, its four projections on $\partial\mathcal{P}$, i.e., p^l, p^r, p^u, and p^d, are *type-1* Steiner points. Clearly, there are $O(n)$ type-1 Steiner points in total.

The *type-2 Steiner points* are on *cut-lines*. In order to facilitate an explanation on our new graph model in Section 3, we organize the cut-lines in a binary tree structure, called the *cut-line tree* and denoted by $T(\mathcal{P})$. The tree $T(\mathcal{P})$ is defined as follows. For each node u of $T(\mathcal{P})$, a set $V(u)$ of obstacle vertices and a cut-line $l(u)$ are associated with u, where $l(u)$ is a vertical line through the median of the x-coordinates of the obstacle vertices in $V(u)$. For the root r of $T(\mathcal{P})$, $V(r)$ is the set of all obstacle vertices of $\mathcal{P}$. For the left (resp., right) child v of u, $V(v)$ consists of the obstacle vertices of $V(u)$ on the left (resp., right) of $l(u)$. Since the number of vertices of $\mathcal{P}$ is n, the height of $T(\mathcal{P})$ is $O(\log n)$. For every node u of $T(\mathcal{P})$, for each vertex $p \in V(u)$, if p is horizontally visible to $l(u)$, then the point $p_h(l(u))$, i.e., the horizontal projection of p on $l(u)$, is a type-2 Steiner point. Since each obstacle vertex defines a type-2 Steiner point on at most one cut-line at each level of $T(\mathcal{P})$, there are $O(n \log n)$ type-2 Steiner points.

The node set of G_{old} consists of all obstacle vertices of $\mathcal{P}$ and all Steiner points thus defined.

The edges of G_{old} are defined as follows. First, for every obstacle vertex p, there is an edge $\overline{pq}$ in G_{old} for each $q \in \{p^l, p^r, p^u, p^d\}$. Second, for every obstacle edge e of $\mathcal{P}$, e may contain multiple type-1 Steiner points, and these Steiner points and the two endpoints of e are the nodes of G_{old} on e; the segment connecting each pair of consecutive graph nodes on e defines an edge in G_{old}. Third, for each cut-line l, any two consecutive type-2 Steiner points on l define an edge in G_{old} if they are visible to each other. Finally, for each obstacle vertex p, if p defines a type-2 Steiner point p' on a cut-line, then $\overline{pp'}$ defines an edge in G_{old}. Clearly, G_{old} has $O(n \log n)$ nodes and $O(n \log n)$ edges.

It was shown in [16, 17] that G_{old} contains a shortest path between any two obstacle vertices. Chen *et al.* [7] used G_{old} to answer two-point queries by "inserting" the query points s and t into G_{old} so that shortest s-t paths are "controlled" by only $O(\log n)$ nodes of G_{old}, called "gateways". The set of gateways of s, denoted by $V_g(s, G_{old})$, is defined as follows. $V_g(s, G_{old})$ consists of two subsets $V_g^1(s, G_{old})$ and $V_g^2(s, G_{old})$. We first define $V_g^1(s, G_{old})$, whose size is $O(1)$. For each $q \in \{s^l, s^r, s^u, s^d\}$, let v_1 and v_2 be the two graph nodes adjacent to q on the obstacle edge containing q; then v_1 and v_2 are in $V_g^1(s, G_{old})$, and the paths $\overline{sq} \cup \overline{qv_1}$ and $\overline{sq} \cup \overline{qv_2}$ are the *gateway edges* from s to v_1 and v_2, respectively. Next, we define $V_g^2(s, G_{old})$, recursively, on the cut-line tree $T(\mathcal{P})$. Let v be the root of $T(\mathcal{P})$. Suppose s is horizontally visible to the cut-line $l(v)$. Let q be the Steiner point on $l(v)$ immediately above (resp., below) the projection point $s_h(l(v))$; if q is visible to $s_h(l(v))$, then q is in $V_g^2(s, G_{old})$ and the path $\overline{ss_h(l(v))} \cup \overline{s_h(l(v))q}$ is the gateway edge from s to q. We also call $l(v)$ a *projection cut-line* of s if s is horizontally visible to $l(v)$. We proceed to the left (resp., right) child of v in $T(\mathcal{P})$ recursively if s is to the left (resp., right) of $l(v)$, until we reach a leaf of $T(\mathcal{P})$. Therefore, $V_g^2(s, G_{old})$ contains $O(\log n)$ type-2 Steiner points on $O(\log n)$ projection cut-lines.

The above defines $V_g(s, G_{old})$, and each gateway $q \in V_g(s, G_{old})$ is associated with a gateway edge between s and q. Henceforth, when we say "a path from s contains a gateway

q", we implicitly mean that the path contains the corresponding gateway edge as well. The above also defines $O(\log n)$ projection cut-lines for s, which will be used later in Section 3. It was shown in [7] that for any obstacle vertex v, there is a shortest s-v path in G_{old} that contains a gateway of s.

Similarly, we define the gateway set $V_g(t, G_{old})$ for t. Assume that there is a shortest s-t path containing an obstacle vertex. Then, there must be a shortest s-t path that contains a gateway $v_s \in V_g(s, G_{old})$, a gateway $v_t \in V_g(t, G_{old})$, and a shortest path from v_s to v_t in the graph G_{old} [7]. Based on this result, a *gateway graph* $G_g(s,t)$ is built for the query on s and t, as follows. The node set of $G_g(s,t)$ is $\{s,t\} \cup V_g(s, G_{old}) \cup V_g(t, G_{old})$. Its edge set consists of all gateway edges and the edges (v_s, v_t) for each $v_s \in V_g(s, G_{old})$ and each $v_t \in V_g(t, G_{old})$, where the weight of (v_s, v_t) is the length of a shortest path from v_s to v_t in G_{old}. Hence, $G_g(s,t)$ has $O(\log n)$ nodes and $O(\log^2 n)$ edges, and if we know the weights of all edges (v_s, v_t), then a shortest s-t path in $G_g(s,t)$ can be found in $O(\log^2 n)$ time. To obtain the weights of all edges (v_s, v_t), we compute a single source shortest path tree in G_{old} from each node of G_{old} in the preprocessing. Then, the weight of each such edge (v_s, v_t) is obtained in $O(1)$ time. Further, suppose we find a shortest s-t path in $G_g(s,t)$ that contains a gateway $v_s \in V_g(s, G_{old})$ and a gateway $v_t \in V_g(t, G_{old})$; then we can report an actual shortest s-t path in time linear in the number of edges of the output path by using the shortest path tree from v_s in G_{old} (which has been computed in the preprocessing).

As discussed in [7], it is possible that no shortest s-t path contains any obstacle vertex. For example, consider a projection point s^r of s and a projection point t^d of t. If $\overline{ss^r}$ intersects $\overline{tt^d}$, say at a point q, then $\overline{sq} \cup \overline{qt}$ is a shortest s-t path; otherwise, if s^r and t^d are both on the same obstacle edge, then $\overline{ss^r} \cup \overline{s^r t^d} \cup \overline{t^d t}$ is a shortest s-t path. We call such shortest s-t paths *trivial shortest paths*. Similarly, trivial shortest s-t paths can also be defined by other projection points in $\{s^l, s^r, s^u, s^d\}$ and $\{t^l, t^r, t^u, t^d\}$. It was shown in [7] that if there is no trivial shortest s-t path, then there exists a shortest s-t path that contains an obstacle vertex. If we know $\{s^l, s^r, s^u, s^d\}$ and $\{t^l, t^r, t^u, t^d\}$, then we can determine whether there exists a trivial shortest s-t path in $O(1)$ time. For any query points s and t, their projection points can be computed easily in $O(\log n)$ time by using the horizontal and vertical visibility decompositions of $\mathcal{P}$, as shown in [7].

3 Reducing the Query Time Based on an Enhanced Graph

In this section, we propose an "enhanced graph" G_E that allows us to reduce the query time to $O(\log n)$. We first define G_E, and then show how to answer two-point queries.

3.1 The Enhanced Graph G_E

First, every node of G_{old} is also a node in G_E. In addition, G_E contains the following *type-3* Steiner points as nodes. To define them, we introduce the concepts of "levels" and "super-levels" on the cut-line tree $T(\mathcal{P})$ defined in Section 2. $T(\mathcal{P})$ has $O(\log n)$ levels. We define the level numbers recursively: The root v is at the first level, and its *level number* is denoted by $ln(v) = 1$; for any node v of $T(\mathcal{P})$, if u is a child of v, then $ln(u) = ln(v) + 1$.

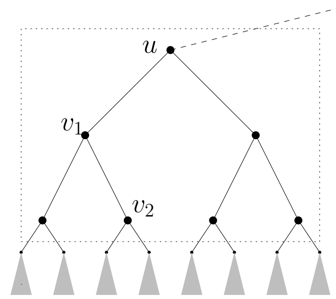

Figure 1: Illustrating $T_u(\mathcal{P})$, i.e., the portion of the tree in the dotted box, where $\sqrt{\log n} = 3$.

We further partition the $O(\log n)$ levels of $T(\mathcal{P})$ into $O(\sqrt{\log n})$ *super-levels*: For any i, $1 \leq i \leq O(\sqrt{\log n})$, the i-th super-level contains the levels from $(i-1) \cdot \sqrt{\log n} + 1$ to $i \cdot \sqrt{\log n}$.

Consider the i-th super-level. Let u be any node at the highest level of this super-level. Let $T_u(\mathcal{P})$ denote the subtree of $T(\mathcal{P})$ rooted at u without including any node outside the i-th super-level (e.g., see Fig. 1 and its corresponding cut-lines and level numbers in Fig. 2). Since $T_u(\mathcal{P})$ has $O(\sqrt{\log n})$ levels, $T_u(\mathcal{P})$ has $O(2^{\sqrt{\log n}})$ nodes. Recall that u is associated with a subset $V(u)$ of obstacle vertices and a vertical cut-line $l(u)$, and for any vertex p in $V(u)$, if p is horizontally visible to $l(u)$, then its projection point $p_h(l(u))$ is a type-2 Steiner point. Each point $p \in V(u)$ defines the following type-3 Steiner points. For each node v in $T_u(\mathcal{P})$, if p is horizontally visible to $l(v)$, then its projection point $p_h(l(v))$ is a type-3 Steiner point (e.g., see Fig. 2; note that if $p \in V(v)$, then the Steiner point is also a type-2 Steiner point). Hence, p defines $O(2^{\sqrt{\log n}})$ type-3 Steiner points in the i-th super-level of $T(\mathcal{P})$. Let $S(p)$ be the set of all type-2 and type-3 Steiner points on the cut-lines of the subtree $T_u(\mathcal{P})$ induced by p, and let $S(p)$ also contain p. In the order of the points in $S(p)$ from left to right, we put an edge in G_E connecting every two consecutive points in $S(p)$ (e.g., see Fig. 2). Since the total number of obstacle vertices in $V(u)$ for all nodes u at the same level of $T(\mathcal{P})$ is n, the number of type-3 Steiner points thus defined in each super-level is $O(n2^{\sqrt{\log n}})$, and the total number of type-3 Steiner points on all cut-lines in $T(\mathcal{P})$ is $O(n\sqrt{\log n}2^{\sqrt{\log n}})$. The number of edges thus added to G_E is also $O(n\sqrt{\log n}2^{\sqrt{\log n}})$.

Hence, the total number of nodes in G_E is $O(n\sqrt{\log n}2^{\sqrt{\log n}})$, which is dominated by the number of type-3 Steiner points. We have also defined above some edges in G_E. Other edges of G_E are defined similarly as in G_{old}. Specifically, first, as in G_{old}, for every obstacle vertex p, there is an edge $\overline{pq}$ in G_E for each $q \in \{p^l, p^r, p^u, p^d\}$. Second, as in G_{old}, for each obstacle edge e, the segment connecting each pair of consecutive graph nodes on e defines an edge in G_E. Third, for each cut-line l, every pair of consecutive Steiner points (type-2 or type-3) on l defines an edge in G_E if these two points are visible to each other. Clearly, the total number of edges in G_E is $O(n\sqrt{\log n}2^{\sqrt{\log n}})$.

This finishes the definition of the graph G_E, which has $O(n\sqrt{\log n}2^{\sqrt{\log n}})$ nodes and $O(n\sqrt{\log n}2^{\sqrt{\log n}})$ edges. The following lemma gives an algorithm for computing G_E.

Lemma 1. *The enhanced graph G_E can be constructed in $O(n \log^{3/2} n 2^{\sqrt{\log n}})$ time.*

Proof. First of all, all type-1 Steiner points are computed easily in $O(n \log n)$ time, e.g., by

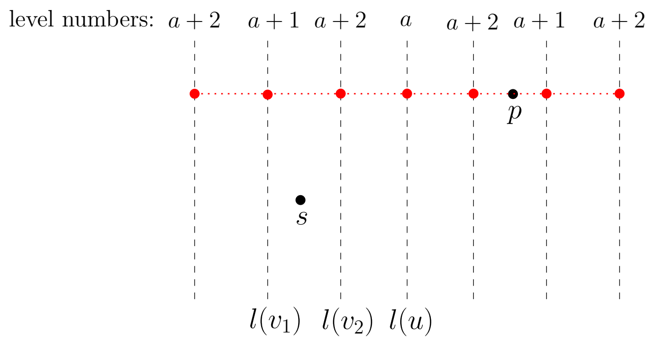

Figure 2: Illustrating the cut-lines and level numbers of the subtree $T_u(\mathcal{P})$ in Fig. 1, where a is the level number $ln(u)$ of the node u. p is an obstacle vertex. If p is visible to all cut-lines, then the red points are type-2 and type-3 Steiner points defined by p and the (red) dotted segments are the corresponding graph edges.

using the vertical and horizontal visibility decompositions of $\mathcal{P}$. The edges of G_E connecting the obstacle vertices and their corresponding type-1 Steiner points can also be computed. For each obstacle edge e, we sort all graph nodes on e and then compute the edges of G_E connecting the consecutive nodes on e. Since there are $O(n)$ type-1 Steiner points, computing these edges takes $O(n \log n)$ time.

Next, we compute both the type-2 and type-3 Steiner points and their adjacent edges. For this, we need to use the two projection points p^l and p^r for each obstacle vertex p of $\mathcal{P}$, which have been computed as type-1 Steiner points. Consider an obstacle vertex p in $V(u)$ for a node u at the highest level of a super-level. For each node v in $T_u(\mathcal{P})$, we need to determine whether p is horizontally visible to $l(v)$, which can be done in $O(1)$ time since p^l and p^r are already known. We also need to have a sorted order of all cut-lines in $T_u(\mathcal{P})$ from left to right, and this ordered list can be obtained by an in-order traversal of $T_u(\mathcal{P})$ in linear time. Therefore, the edges of G_E connecting the Steiner points defined by p on consecutive cut-lines in this super-level can be computed in time linear in the number of nodes in $T_u(\mathcal{P})$. Since there are $O(n\sqrt{\log n}2^{\sqrt{\log n}})$ type-2 and type-3 Steiner points, computing all such edges takes $O(n\sqrt{\log n}2^{\sqrt{\log n}})$ time.

It remains to compute the graph edges on all cut-lines connecting consecutive Steiner points (if they are visible to each other). This step is done in $O(n \log^{3/2} n 2^{\sqrt{\log n}})$ time by a sweeping algorithm, as follows. For each cut-line l, we sort the Steiner points on l by their y-coordinates, and determine whether every two consecutive Steiner points on l are visible to each other. For this, we sweep a vertical line L from left to right. During the sweeping, we use a balanced binary search tree T to maintain the maximal intervals of L that are in the free space of $\mathcal{P}$ (there are $O(n)$ such intervals). At each obstacle vertex, we update T in $O(\log n)$ time. At each (vertical) cut-line l, for every two consecutive Steiner points, we determine whether they are visible to each other in $O(\log n)$ time by checking whether they are in the same maximal interval maintained by T. Since there are $O(n\sqrt{\log n}2^{\sqrt{\log n}})$ pairs of consecutive Steiner points on all cut-lines, computing all edges of G_E on the cut-lines takes $O(n \log^{3/2} n 2^{\sqrt{\log n}})$ time.

In summary, the graph G_E can be computed in $O(n \log^{3/2} n 2^{\sqrt{\log n}})$ time. $\square$

3.2 Reducing the Query Time

We use the enhanced graph G_E to reduce the query time to $O(\log n)$. Consider two query points s and t. The key idea is as follows. We define a new set of gateways for s, denoted by $V_g(s, G_E)$, which contains $O(\sqrt{\log n})$ nodes of G_E, such that for any obstacle vertex p of $\mathcal{P}$, there exists a shortest path from s to p through a gateway of $V_g(s, G_E)$. The set $V_g(s, G_E)$ can be partitioned into two subsets $V_g^1(s, G_E)$ and $V_g^2(s, G_E)$, where $V_g^1(s, G_E)$ (of size $O(1)$) is exactly the same as $V_g^1(s, G_{old})$ defined on G_{old} in Section 2. Below, we define the subset $V_g^2(s, G_E)$.

Recall that s has $O(\log n)$ projection cut-lines, as defined in Section 2. By definition, s is horizontally visible to all its projection cut-lines. Since G_E has more Steiner points than G_{old}, the intuition is that we do not have to include gateways in each projection cut-line of s. More specifically, we only need to include gateways in two projection cut-lines in each super-level (one to the left of s and the other to the right of s). The details are given below.

We define the *relevant projection cut-lines* of s, as follows. Let S be the set of projection cut-lines of s to the right of s. Consider a cut-line $l \in S$ and suppose l is associated with a node u in the i-th super-level of the cut-line tree $T(\mathcal{P})$ for some i. Then l is a *relevant projection cut-line* of s if $ln(u) > ln(v)$ (i.e., their level numbers) for every node v with $v \neq u$ in the i-th super-level of $T(\mathcal{P})$ such that the cut-line $l(v)$ of v is also in S. In other words, $l(u)$ is a relevant projection cut-line of s if u has the largest distance in $T(\mathcal{P})$ from the root among all nodes v in the i-th super-level of $T(\mathcal{P})$ whose cut-lines $l(v)$ are in S. For example, in Fig. 1 and Fig. 2, suppose s is between the cut-lines $l(v_1)$ and $l(v_2)$ and both $l(u)$ and $l(v_2)$ are horizontally visible to s; then among the cut-lines of all nodes in $T_u(\mathcal{P})$, only $l(v_2)$ and $l(u)$ are in S, but only $l(v_2)$ is the relevant projection cut-line of s. The relevant projection cut-lines of s to the left of s are defined similarly. Since s has $O(\log n)$ projection cut-lines and any two of them are at different levels of $T(\mathcal{P})$, the number of relevant projection cut-lines of s is $O(\sqrt{\log n})$, i.e., at most two from each super-level of $T(\mathcal{P})$ (one to the left of s and the other to the right of s). For each relevant projection cut-line l of s, the Steiner point p (if any) immediately above (resp., below) the projection point $s_h(l)$ of s on l is in $V_g^2(s, G_E)$ if p is visible to $s_h(l)$. Thus, $|V_g^2(s, G_E)| = O(\sqrt{\log n})$.

Clearly, the size of $V_g(s, G_E)$ is $O(\sqrt{\log n})$. We also define the gateway edge for each gateway of $V_g(s, G_E)$ and s in the same way as in Section 2. Below, when we say a shortest path from s containing a gateway, we mean the path containing the corresponding gateway edge as well.

Lemma 2. *For any obstacle vertex p of $\mathcal{P}$, there exists a shortest path from s to p using G_E that contains a gateway of s in $V_g(s, G_E)$.*

Proof. Recall that $V_g(s, G_{old})$ is the gateway set of s defined on G_{old} in Section 2, and by [7], there exists a shortest path $\pi(s, p)$ from s to p using G_{old} that contains a point $a \in V_g(s, G_{old})$.

By the definition of G_E, if any edge e of G_{old} connecting two nodes u and v is not an edge of G_E, then e is path in G_E. Hence, $\pi(s, p)$ is still a shortest path in G_E. For any point $a \in V_g(s, G_{old})$ that is on a shortest s-p path, we call it a *via point*. If any via point

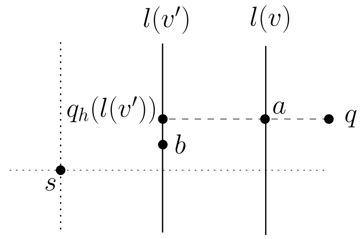

Figure 3: Illustrating the proof of Lemma 2: q is the obstacle vertex that defines the Steiner point a; $l(v')$ is between s and $l(v)$.

a is in $V_g^1(s, G_{old})$, then a is in $V_g(s, G_E)$ since $V_g^1(s, G_E) = V_g^1(s, G_{old})$, and we are done. Otherwise, all via points must be in $V_g^2(s, G_{old})$. If any such via point $a \in V_g^2(s, G_{old})$ is also in $V_g^2(s, G_E)$, then we are done as well. It remains to prove the case where for each via point a, we have $a \in V_g^2(s, G_{old})$ and $a \notin V_g^2(s, G_E)$. Recall that every node of G_{old}, including each via point a, is also a node of G_E. Below, we find an xy-monotone path from s to such a via point a along G_E that contains a gateway $b \in V_g^2(s, G_E)$. Since any xy-monotone path is a shortest path, this gives a shortest s-p path (through a) containing a gateway b of s in $V_g(s, G_E)$, thus proving the lemma.

Without loss of generality, we assume that a is to the right of s and above s (i.e., a is to the northeast of s, see Fig. 3). Suppose a is on the cut-line $l(v)$ of a node v in the i-th super-level of $T(\mathcal{P})$. If $l(v)$ is a relevant cut-line of s, then there must be a gateway b of s in $V_g^2(s, G_E)$ lying in the vertical segment $\overline{s_h(l(v))a}$ on $l(v)$ (possibly $b = a$), and thus we are done. Otherwise, $l(v)$ is not a relevant cut-line of s, and there exists a relevant cut-line $l(v')$ of s to the right of s such that v' is in the i-th super-level of $T(\mathcal{P})$ and $ln(v') > ln(v)$. Next, we show that b lies on $l(v')$.

It was shown in [7] (Lemma 3.4) that the level numbers of the projection cut-lines of s to the right of s, in the left-to-right order, are decreasing. This observation can also be seen easily by considering the projection cut-lines of $T(\mathcal{P})$ in a top-down manner. Hence, $l(v')$ is to the left of $l(v)$ (see Fig. 3). Let q be the obstacle vertex that defines the Steiner point a on $l(v)$. By our definition of Steiner points, q must be in $V(u)$ for the node u that is the highest ancestor of v (and v') in the i-th super-level. Therefore, if q is horizontally visible to $l(v')$, then q also defines a Steiner point on $l(v')$. We now show that q is horizontally visible to $l(v')$, and for this, it suffices to prove that a is horizontally visible to $l(v')$ since q is horizontally visible to a. Because $a \in V_g^2(s, G_{old})$ and no via point is in $V_g^1(s, G_{old})$, it was shown in [7] that a must be horizontally visible to the vertical line through s. Since $l(v')$ is between s and $a \in l(v)$, a is also horizontally visible to $l(v')$.

Thus, q defines a Steiner point on $l(v')$, i.e., the point $q_h(l(v'))$ (see Fig. 3). By the definition of $V_g^2(s, G_E)$, the lowest Steiner point b on $l(v')$ above s must be a gateway in $V_g^2(s, G_E)$. Note that b may or may not be $q_h(l(v'))$, but b cannot be higher than $q_h(l(v'))$. Thus, the concatenation of the gateway edge from s to b, $\overline{bq_h(l(v'))}$, and $\overline{q_h(l(v'))a}$, which is an xy-monotone path from s to a using G_E, contains the gateway b of $V_g^2(s, G_E)$. The lemma thus follows. □

Similarly, we define the gateway set $V_g(t, G_E)$ for t in G_E. The similar result for t

as Lemma 2 for s also holds. Thus, we have the following corollary.

Corollary 1. *If there exists a shortest s-t path through an obstacle vertex of $\mathcal{P}$, then there exists a shortest s-t path through a gateway of s in $V_g(s, G_E)$ and a gateway of t in $V_g(t, G_E)$.*

Next, we give an algorithm for computing the two gateway sets $V_g(s, G_E)$ and $V_g(t, G_E)$.

Lemma 3. *With a preprocessing of $O(n \log^{3/2} n 2^{\sqrt{\log n}})$ time and $O(n\sqrt{\log n}2^{\sqrt{\log n}})$ space, we can compute the gateway sets $V_g(s, G_E)$ and $V_g(t, G_E)$ in $O(\log n)$ time for any query points s and t.*

Proof. We only discuss the case for computing $V_g(s, G_E)$ since $V_g(t, G_E)$ can be computed similarly. To compute the set $V_g^1(s, G_E)$, it suffices to determine the four projection points $\{s^l, s^r, s^u, s^d\}$ of s on $\partial\mathcal{P}$, which can be computed in $O(\log n)$ time by using the horizontal and vertical visibility decompositions of $\mathcal{P}$. These two visibility decompositions can be built in $O(n \log n)$ time by standard sweeping algorithms. After that, we also need to build a point location data structure [18, 32] on each of the two decompositions in additional $O(n)$ time.

To compute the set $V_g^2(s, G_E)$, it might be possible to modify the approach in [7]. However, to explain the approach in [7], we may have to review a number of observations given in [7]. To avoid a tedious discussion, we propose the following algorithm that is simple.

We first obtain the set S of all relevant projection cut-lines of s. This can be done in $O(\log n)$ time by following the cut-line tree $T(\mathcal{P})$ from the root and using s^l and s^r to determine the horizontal visibility of s. Note that the cut-lines of S are at some nodes on a path from the root to a leaf. To obtain $V_g^2(s, G_E)$, for each cut-line $l \in S$, we need to: (1) find the Steiner point p on l immediately above (resp., below) $s_h(l)$, and (2) determine whether p is visible to $s_h(l)$.

Consider a cut-line $l \in S$. Let $v_1(l)$ and $v_2(l)$ be the two gateways of $V_g^2(s, G_E)$ on l (if any) such that $v_1(l)$ is above $v_2(l)$. That is, $v_1(l)$ (resp., $v_2(l)$) is the Steiner point on l immediately above (resp., below) $s_h(l)$ and visible to $s_h(l)$. If we maintain a sorted list of all Steiner points on l, then $v_1(l)$ and $v_2(l)$ can be found by binary search on the sorted list. However, there are two issues with this approach. First, if we do binary search on each cut-line of S, since $|S| = O(\sqrt{\log n})$, it takes $O(\log^{3/2} n)$ time on all cut-lines of S. Second, even if we find $v_1(l)$ and $v_2(l)$, we still need to check whether $s_h(l)$ is visible to them. To resolve these two issues, we take the following approach.

For every Steiner point p on the cut-line l, suppose we associate with p its upward and downward projection points p_u and p_d on $\partial\mathcal{P}$. Then once we find the Steiner point q on l immediately above (resp., below) $s_h(l)$, we can determine easily whether q is visible to $s_h(l)$ using q_u and q_d; if q is visible to $s_h(l)$, then $v_1(l) = q$ (resp., $v_2(l) = q$), or else $v_1(l)$ (resp., $v_2(l)$) does not exist. For any Steiner point p on l, p^l and p^r can be found in $O(\log n)$ time by using the vertical visibility decomposition of $\mathcal{P}$. Since there are $O(n\sqrt{\log n}2^{\sqrt{\log n}})$ Steiner points p on all cut-lines of $T(\mathcal{P})$, their projection points p^u and p^d can be computed in totally $O(n \log^{3/2} n 2^{\sqrt{\log n}})$ time.

Next, for each cut-line l, we sort all Steiner points on l. With this, one can compute all gateways of $V_g^2(s, G_E)$ in $O(\log^2 n)$ time by doing binary search on each relevant projection cut-line of s. To reduce the query time to $O(\log n)$, we make use of the fact that all relevant projection cut-lines of s are at the nodes on a path of $T(\mathcal{P})$ from the root to a leaf. We build a fractional cascading structure [5] on the sorted lists of Steiner points on all cut-lines along $T(\mathcal{P})$, such that the searches on all cut-lines at the nodes on any path of $T(\mathcal{P})$ from the root to a leaf take $O(\log n)$ time. Hence, all gateways of $V_g^2(s, G_E)$ can be computed in $O(\log n)$ time. Since the total number of Steiner points in the sorted lists of all cut-lines of $T(\mathcal{P})$ is $O(n\sqrt{\log n}2^{\sqrt{\log n}})$, the fractional cascading structure can be built in $O(n\sqrt{\log n}2^{\sqrt{\log n}})$ space and $O(n \log^{3/2} n 2^{\sqrt{\log n}})$ time. The lemma thus follows. □

Theorem 1. *We can build a data structure of size $O(n^2 \cdot \log n \cdot 4^{\sqrt{\log n}})$ in $O(n^2 \cdot \log^2 n \cdot 4^{\sqrt{\log n}})$ time such that each two-point L_1 shortest path query can be answered in $O(\log n)$ time (i.e., for any two query points s and t, the length of a shortest s-t path can be found in $O(\log n)$ time and an actual path can be reported in additional time linear in the number of edges of the output path).*

Proof. In the preprocessing, we first build the graph G_E. Then, for each node v of G_E, we compute a shortest path tree in G_E from v. We also maintain a shortest path length table such that for any two nodes u and v, the shortest u-v path length in G_E can be obtained in $O(1)$ time. Since G_E is of a size $O(n\sqrt{\log n}2^{\sqrt{\log n}})$, computing and maintaining all these shortest path trees in G_E take $O(n^2 \log n 4^{\sqrt{\log n}})$ space and $O(n^2 \log^2 n 4^{\sqrt{\log n}})$ time. We also do the preprocessing for Lemma 3.

Given any two query points s and t, we first check whether there is a trivial shortest s-t path, as discussed in Section 2, in $O(\log n)$ time by using the algorithm in [7] (with an $O(n \log n)$ time preprocessing). If there is a trivial shortest s-t path, then we are done. Otherwise, there must be a shortest s-t path that contains an obstacle vertex of $\mathcal{P}$. Then, we first compute the gateway sets $V_g(s, G_E)$ and $V_g(t, G_E)$ in $O(\log n)$ time by Lemma 3. Finally, we determine the shortest s-t path length by using the gateway graph as discussed in Section 2, in $O(\log n)$ time, since there are $O(\sqrt{\log n})$ gateways and thus the gateway graph has $O(\sqrt{\log n})$ nodes and $O(\log n)$ edges.

We can report an actual shortest s-t path in additional time linear in the number of edges of the output path by using the shortest path trees of G_E. This proves the theorem. □

4 Reducing the Time and Space Bounds of the Preprocessing

In this section, we improve the preprocessing in Theorem 1 to $O(n + h^2 \cdot \log h \cdot 4^{\sqrt{\log h}})$ space and $O(n + h^2 \cdot \log^2 h \cdot 4^{\sqrt{\log h}})$ time, while maintaining the $O(\log n)$ query time. For this, we shall make use of the extended corridor data structure [8, 9, 11, 31], and more importantly, explore a number of new observations, which may be interesting in their own right.

The corridor structure has been used to solve shortest path problems (e.g., [27, 30, 31]), and new concepts like "ocean", "bays", and "canals" have been introduced [8, 9, 10, 12, 11, 13], which we refer to as the "extended corridor structure". This structure is a subdivision of the free space on which algorithms for specific problems rely. While the extended corridor structure itself is relatively simple, the main difficulty is to design efficient algorithms to exploit it. In some sense, the role played by the extended corridor structure is similar to that of triangulations for many geometric algorithms. We briefly review it in Section 4.1, since our presentation uses many notations introduced in it.

4.1 The Extended Corridor Structure

For simplicity of discussion, we assume that the obstacles of $\mathcal{P}$ are all contained in a rectangle $\mathcal{R}$. Let $\mathcal{F}$ denote the free space in $\mathcal{R}$, and $\mathit{Tri}(\mathcal{F})$ denote a triangulation of $\mathcal{F}$ (see Fig. 4). The line segments of $\mathit{Tri}(\mathcal{F})$ that are not obstacle edges are referred to as *diagonals*.

Let $G(\mathcal{F})$ denote the dual graph of $\mathit{Tri}(\mathcal{F})$, i.e., each node of $G(\mathcal{F})$ corresponds to a triangle of $\mathit{Tri}(\mathcal{F})$ and each edge connects two nodes corresponding to two triangles sharing a diagonal of $\mathit{Tri}(\mathcal{F})$. 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 (see Fig. 4), which has $O(h)$ faces, nodes, and edges [31]. Every node of G^3 corresponds to a triangle in $\mathit{Tri}(\mathcal{F})$, called a *junction triangle* (see Fig. 4). 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.

The boundary of each corridor C consists of four parts (see Fig. 5): (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, and the two boundary portions defined above in (1) and (3) are two *sides* of C. 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 $\pi(a,b), \pi(e,f)$, $\overline{be}$, and $\overline{fa}$ is called an *hourglass*, which is *open* if $\pi(a,b) \cap \pi(e,f) = \emptyset$ and *closed* otherwise. If H_C is open, then both $\pi(a,b)$ and $\pi(e,f)$ are convex chains and are called the *sides* of H_C; otherwise, H_C consists of two "funnel" and a path $\pi_C = \pi(a,b) \cap \pi(e,f)$ joining the two apices of the two funnels, and π_C is called the *corridor path* of C. The two funnel apices (e.g., x and y in Fig. 5) are called *corridor path terminals*. Each side of a funnel is also a convex chain.

Let $\mathcal{M}$ be the union of the $O(h)$ junction triangles, open hourglasses, and funnels. Then $\mathcal{M} \subseteq \mathcal{F}$. We call $\mathcal{M}$ the *ocean*. Since the sides of open hourglasses and funnels are all convex, the boundary $\partial\mathcal{M}$ of $\mathcal{M}$ consists of $O(h)$ convex chains with a total of $O(n)$ vertices; also, there are $O(h)$ reflex vertices on $\partial\mathcal{M}$, which are corridor path terminals. We further partition the free space $\mathcal{F} \setminus \mathcal{M}$ into regions called *bays* and *canals*, as follows.

Consider the hourglass H_C of a corridor C. If H_C is open, then H_C has two sides. Let $S_1(H_C)$ be one side of H_C. The obstacle vertices on $S_1(H_C)$ all lie on the same side of C. Let c and d be any two consecutive vertices on $S_1(H_C)$ such that $\overline{cd}$ is not an edge of C (e.g., see the left figure in Fig. 5). The free region enclosed by $\overline{cd}$ and the boundary portion of C between c and d is called a *bay*, denoted by $\mathit{bay}(\overline{cd})$. We call $\overline{cd}$ the *gate* of $\mathit{bay}(\overline{cd})$,

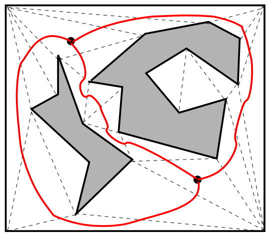

Figure 4: [9, 10] Illustrating a triangulation of the free space among two obstacles and the corridors (indicated by red solid curves). There are two junction triangles marked by a large dot inside each of them, connected by three solid (red) curves. Removing the two junction triangles results in three corridors.

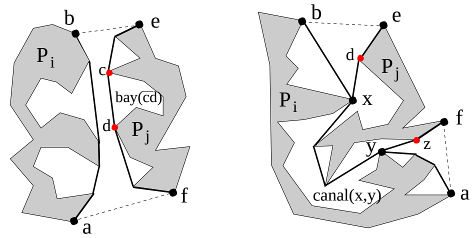

Figure 5: [9, 10] Illustrating an open hourglass (left) and a closed hourglass (right) with a corridor path connecting the apices x and y of the two funnels. The dashed segments are diagonals. The paths $\pi(a, b)$ and $\pi(e, f)$ are shown with thick solid curves. A bay $bay(\overline{cd})$ with gate $\overline{cd}$ (left) and a canal $canal(x, y)$ with gates $\overline{xd}$ and $\overline{yz}$ (right) are also indicated.

which is an edge shared by $bay(\overline{cd})$ and $\mathcal{M}$. If H_C is closed, let x and y be the two apices of its two funnels. Consider two consecutive vertices c and d on a side of any funnel such that $\overline{cd}$ is not an obstacle edge. If neither c nor d is a funnel apex, then c and d must lie on the same side of C and the segment $\overline{cd}$ also defines a bay. However, if c or d is a funnel apex (say, $c = x$), then c and d may lie on different sides of C. If they lie on the same side of C, then they also define a bay; otherwise, we call $\overline{xd}$ the *canal gate* at $x = c$ (see Fig. 5). Similarly, there is a canal gate at the other funnel apex y, say $\overline{yz}$. The sub-region of C between $\overline{xd}$ and $\overline{yz}$ that contains the corridor path of H_C is called a *canal*, denoted by $canal(x, y)$.

Every bay or canal is a simple polygon. The ocean, bays, and canals together constitute the free space $\mathcal{F}$. While the total number of all bays is $O(n)$, the total number of all canals is $O(h)$.

4.2 Queries in the Ocean $\mathcal{M}$

For any two points s and t in the ocean $\mathcal{M}$, it has been proved that there exists an L_1 shortest s-t path in the free space of the union of $\mathcal{M}$ and all corridor paths [8, 9, 11]. Let $\mathcal{M}'$ be the union of $\mathcal{M}$ and all corridor paths. Thus, if s and t are both in $\mathcal{M}$, then there is a shortest s-t path in $\mathcal{M}'$.

In this subsection, we will first construct a graph $G_E(\mathcal{M})$ of size $O(h \cdot \sqrt{\log h} \cdot 2^{\sqrt{\log h}})$ on $\mathcal{M}$, in a similar fashion as G_E in Section 3. Using the graph $G_E(\mathcal{M})$ and with additional $O(n)$ space, for any query points s and t in $\mathcal{M}$, the shortest path query can be answered in $O(\log n)$ time.

Recall that $\mathcal{R}$ is the rectangle that contains all obstacles of $\mathcal{P}$. Let $\mathcal{Q} = \mathcal{R} \setminus \mathcal{M}$. Note that $\partial\mathcal{Q}$ is $\partial\mathcal{M}$. Hence, $\partial\mathcal{Q}$ consists of $O(h)$ convex chains with a total of $O(n)$ vertices, and $\partial\mathcal{Q}$ also contains $O(h)$ reflex vertices that are corridor path terminals. Since $\mathcal{P}$ has h obstacles, $\mathcal{Q}$ contains at most h connected components and each obstacle of $\mathcal{P}$ is contained in a component of $\mathcal{Q}$. For any point q in $\mathcal{M}$, in this subsection, let q^l, q^r, q^u, and q^d denote the leftward, rightward, upward, and downward projection points of q on $\partial\mathcal{Q}$, respectively.

An obstacle vertex p on $\partial\mathcal{Q}$ is said to be *extreme* if both its incident edges on $\partial\mathcal{Q}$

are on the same side of the vertical or horizontal line through p. Let $V_e(\mathcal{Q})$ denote the set of all extreme vertices and corridor path terminals of $\mathcal{Q}$. Since $\partial\mathcal{Q}$ consists of $O(h)$ convex chains and $O(h)$ reflex vertices that are corridor path terminals, $|V_e(\mathcal{Q})| = O(h)$. We could build a graph on $V_e(\mathcal{Q})$ with respect to $\mathcal{Q}$ in a similar way as we built G_E on the obstacle vertices of $\mathcal{P}$ in Section 3, and then use this graph to answer queries when both query points are in $\mathcal{M}$. However, in order to handle the general queries (in Section 4.3) for which at least one query point is not in $\mathcal{M}$, we need to consider more points for building the graph. Specifically, let $\mathcal{V}(\mathcal{Q}) = \{p^l, p^r, p^u, p^d \mid p \in V_e(\mathcal{Q})\} \cup V_e(\mathcal{Q})$, i.e., in addition to $V_e(\mathcal{Q})$, $\mathcal{V}(\mathcal{Q})$ also contains the four projections of all points in $V_e(\mathcal{Q})$ on $\partial\mathcal{Q}$. Since $|V_e(\mathcal{Q})| = O(h)$, we have $|\mathcal{V}(\mathcal{Q})| = O(h)$.

For each connected component Q of $\mathcal{Q}$, let $\mathcal{V}(Q)$ denote the set of points of $\mathcal{V}(\mathcal{Q})$ on Q. Consider any two points a and b of $\mathcal{V}(Q)$ that are consecutive on the boundary ∂Q of Q. By the definition of a and b, the boundary portion of ∂Q between a and b that contains no other points of $\mathcal{V}(Q)$ must be an xy-monotone path (similar results were also given in [8, 9, 11, 27]), and we call it an *elementary curve* of ∂Q. Hence, for any two points on an elementary curve, the portion of the curve between the two points is a shortest path between the two points.

Our goal is to build a graph, denoted by $G_E(\mathcal{M})$, on $\mathcal{V}(\mathcal{Q})$ with respect to $\mathcal{Q}$ in a similar way as we built G_E in Section 3, and use it to answer queries. To argue the correctness of our approach, we also define a graph $G_{old}(\mathcal{M})$ on $\mathcal{V}(\mathcal{Q})$ and $\mathcal{Q}$ in a similar way as G_{old} on $\mathcal{P}$. Again, $G_{old}(\mathcal{M})$ is only for showing the correctness of our approach based on $G_E(\mathcal{M})$ (recall that we use G_{old} to show the correctness of using G_E). Below, we define $G_E(\mathcal{M})$ and $G_{old}(\mathcal{M})$ simultaneously.

We first define their node sets. Each point of $\mathcal{V}(\mathcal{Q})$ defines a node in both graphs. In addition, $G_{old}(\mathcal{M})$ has type-1 and type-2 Steiner points as nodes; $G_E(\mathcal{M})$ has type-1, type-2, and type-3 Steiner points as nodes. Such Steiner points are defined using $\mathcal{V}(\mathcal{Q})$ in a similar way as before, but with respect to $\partial\mathcal{Q}$. Specifically, for each point $p \in \mathcal{V}(\mathcal{Q})$, its four projections p^l, p^r, p^u, and p^d on $\partial\mathcal{Q}$ are type-1 Steiner points. Let $T(\mathcal{M})$ be the cut-line tree defined on the points of $\mathcal{V}(\mathcal{Q})$, similar to $T(\mathcal{P})$. Each node u of $T(\mathcal{M})$ is associated with a subset $V(u) \subseteq \mathcal{V}(\mathcal{Q})$ and a vertical cut-line $l(u)$ through the median of the x-coordinates of the points in $V(u)$. Since $|\mathcal{V}(\mathcal{Q})| = O(h)$, $T(\mathcal{M})$ has $O(\log h)$ levels and $O(\sqrt{\log h})$ super-levels. For every node $u \in T(\mathcal{M})$, for each point $p \in V(u)$, if p is horizontally visible to $l(u)$, then the projection of p on $l(u)$ is a type-2 Steiner point. Also, there are $O(h\sqrt{\log h}2^{\sqrt{\log h}})$ type-3 Steiner points on the cut-lines of $T(\mathcal{M})$, which are defined in a similar way as in Section 3, and we omit the details.

The edge sets of the two graphs are defined similarly as those in G_{old} and G_E. We only point out the differences here. The first difference is that for each corridor path, since its two terminals define two nodes in $G_{old}(\mathcal{M})$ (resp., $G_E(\mathcal{M})$), we define an edge in $G_{old}(\mathcal{M})$ (resp., $G_E(\mathcal{M})$) that connects these two nodes and the weight of the edge is the length of the corridor path. The second difference is as follows. In G_{old} and G_E, for each obstacle edge e of $\mathcal{P}$, both graphs have an edge connecting each pair of consecutive graph nodes on e. In contrast, here we consider each individual elementary curve of $\mathcal{Q}$ instead of each individual edge of $\mathcal{Q}$ because not every vertex of $\mathcal{Q}$ defines a node in $G_{old}(\mathcal{M})$ and $G_E(\mathcal{M})$.

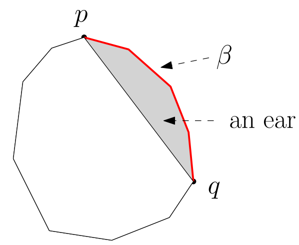

Figure 6: Illustrating an ear bounded by $\overline{pq}$ and an elementary curve β.

Specifically, consider each elementary curve β of $\mathcal{Q}$. Note that the two endpoints of β must be in $\mathcal{V}(Q)$ and thus define two nodes in both graphs. For each pair of consecutive graph nodes along β, we put an edge in both $G_{old}(\mathcal{M})$ and $G_E(\mathcal{M})$ whose weight is the length of the portion of β between these two points. We then have the following lemma.

Lemma 4. *For any two points u and v in $\mathcal{V}(\mathcal{Q})$, a shortest path from u to v in $G_{old}(\mathcal{M})$ corresponds to a shortest path from u to v in the plane, and a shortest path from u to v in $G_E(\mathcal{M})$ also corresponds to a shortest path from u to v in the plane.*

Proof. We first show that a shortest path from u to v in $G_{old}(\mathcal{M})$ corresponds to a shortest path from u to v in the plane, and then show a shortest path from u to v in $G_E(\mathcal{M})$ corresponds to a shortest path from u to v in $G_{old}(\mathcal{M})$. This will prove the lemma.

To show a shortest path from u to v in $G_{old}(\mathcal{M})$ corresponds to a shortest path from u to v in the plane, we will build a new graph G and prove the following: (1) a shortest path from u to v in G corresponds to a shortest path from u to v in $G_{old}(\mathcal{M})$, and (2) a shortest path from u to v in G corresponds to a shortest path from u to v in the plane. Below, to define the graph G, we first review some observations that have been discovered in the previous work.

Let Q be any connected component of $\mathcal{Q}$. Consider an elementary curve β of Q with endpoints p and q. By the definition of elementary curves, the line segment $\overline{pq}$ must be inside Q (similar results were given in [8, 9, 11]). We call the region enclosed by β and $\overline{pq}$ an *ear* of Q (e.g., see Fig. 6), $\overline{pq}$ the *base* of the ear, and β the elementary curve of the ear. It is possible that β is $\overline{pq}$, in which case the ear is $\overline{pq}$. It is easy to see that the bases of all elementary curves of Q do not intersect except at their endpoints [8, 9, 11]. Hence, if we connect the bases of its elementary curves, we obtain a simple polygon that is contained in Q; we call this simple polygon the *core* of Q, denoted by Q_{core}. Clearly, the union of Q_{core} and all the ears of Q is Q. Denote by $\mathcal{Q}_{core}$ the set of cores of all components of $\mathcal{Q}$. Note that the vertex set of $\mathcal{Q}_{core}$ is $\mathcal{V}(\mathcal{Q})$ and the edges of $\mathcal{Q}_{core}$ are the bases of all ears of $\mathcal{Q}$. Thus, $\mathcal{Q}_{core}$ has $O(h)$ vertices and edges. By the results in [8, 9, 11], for any two points in $\mathcal{M}$, in particular, any two vertices u and v in $\mathcal{V}(\mathcal{Q})$, there is a shortest u-v path in the plane that avoids all cores of $\mathcal{Q}_{core}$ and possibly contains corridor paths. More specifically, there exists a shortest path $\pi(u, v)$ from u to v that contains a sequence of vertices of $\mathcal{V}(\mathcal{Q})$, $p_1, p_2, \ldots, p_k$, in this order, with $u = p_1$ and $v = p_k$, such that for any two consecutive vertices p_i and p_{i+1}, $1 \leq i \leq k-1$, if p_i and p_{i+1} are terminals of the same corridor path, then the entire corridor path is contained in $\pi(u, v)$, or else $\pi(u, v)$ contains the line segment $\overline{p_i p_{i+1}}$ which does not intersect the interior of any core in $\mathcal{Q}_{core}$.

We build a graph $G = G_{old}(\mathcal{Q}_{core})$ on $\mathcal{V}(\mathcal{Q})$ with respect to the cores of $\mathcal{Q}$, in the same way as G_{old} on $\mathcal{P}$ in [7, 16, 17], with the only difference that if two nodes of G are terminals of the same corridor path, then there is an extra edge in G connecting these two nodes whose weight is the length of the corridor path. Note that u and v define two nodes in G. Based on the above discussion, we claim that the shortest path $\pi(u, v)$ defined above must correspond to a shortest path from u to v in G. Indeed, for any i, $1 \leq i \leq k-1$, if p_i and p_{i+1} are terminals of the same corridor path, then recall that $\pi(u, v)$ contains the entire corridor path and there is an edge in G connecting p_i and p_{i+1} whose weight is the length of that corridor path; otherwise, $\pi(u, v)$ contains the segment $\overline{p_i p_{i+1}}$, and by the proof in [16, 17], there must be a path in G whose length is equal to that of $\overline{p_i p_{i+1}}$ since p_i is visible to p_{i+1} with respect to the cores of $\mathcal{Q}$. This proves that there is a shortest u-v path in G whose length is equal to that of $\pi(u, v)$.

Next, we prove that a shortest u-v path in G must correspond to a shortest u-v path in $G_{old}(\mathcal{M})$. To make the paper more self-contained, we give some details below; refer to [27, 8, 9] for more details.

Both $G_{old}(\mathcal{M})$ and G are built on $\mathcal{V}(\mathcal{Q})$ in the same way, with the only difference that $G_{old}(\mathcal{M})$ is built with respect to $\mathcal{Q}$ while G is built with respect to $\mathcal{Q}_{core}$. A useful fact is that for any two points a and b on any elementary curve β, the length of the portion of β between a and b is equal to that of the segment $\overline{ab}$ because β is xy-monotone. Note that the space outside $\mathcal{Q}_{core}$ is the union of the space outside $\mathcal{Q}$ and all ears of $\mathcal{Q}$. Since both graphs have extra edges to connect corridor path terminals, to prove that a shortest u-v path in G corresponds to a shortest u-v path in $G_{old}(\mathcal{M})$, based on the analysis in [16, 17], we only need to show the following: For any two vertices a and b of $\mathcal{V}(\mathcal{Q})$ visible to each other with respect to $\mathcal{Q}_{core}$ such that no other vertices of $\mathcal{V}(\mathcal{Q})$ than a and b are in the axis-parallel rectangle $R(a, b)$ that has $\overline{ab}$ as a diagonal, there must be an xy-monotone path between a and b in $G_{old}(\mathcal{M})$. Note that a may not be visible to b with respect to $\mathcal{Q}$.

By the construction of the graph G [16, 17], there must be an xy-monotone path from a to b in G, for which there are two possible cases. Below, we prove in each case there is also an xy-monotone path from a to b in $G_{old}(\mathcal{M})$. Without loss of generality, we assume b is to the northeast of a.

Case 1 If any core of $\mathcal{Q}_{core}$ intersects the interior of the rectangle $R(a, b)$, then as shown in [16, 17], either the rightward projection of a on $\partial\mathcal{Q}_{core}$ and the downward projection of b on $\partial\mathcal{Q}_{core}$ are both on the same edge of $\partial\mathcal{Q}_{core}$ that intersects $R(a, b)$ (e.g., see Fig. 7), or the upward projection of a on $\partial\mathcal{Q}_{core}$ and the leftward projection of b on $\partial\mathcal{Q}_{core}$ are both on the same edge of $\partial\mathcal{Q}_{core}$ that intersects $R(a, b)$. Here, we assume that the former case occurs. Let a_1 be the rightward projection of a on $\partial\mathcal{Q}_{core}$ and b_1 be the downward projection of b on $\partial\mathcal{Q}_{core}$, and $\overline{a_2 b_2}$ be the edge of $\mathcal{Q}_{core}$ that contains both a_1 and b_1. By the construction of G, there is an xy-monotone path from a to b consisting of $\overline{aa_1} \cup \overline{a_1 b_1} \cup \overline{b_1 b}$. Below, we show that there is also an xy-monotone path from a to b in $G_{old}(\mathcal{M})$.

Let $ear(\overline{a_2 b_2})$ be the ear of $\mathcal{Q}$ whose base is $\overline{a_2 b_2}$. Let β be the elementary curve of $ear(\overline{a_2 b_2})$. Since no vertex of $\mathcal{V}(\mathcal{Q}) - \{a, b\}$ is in $R(a, b)$ and all extreme points of $\mathcal{Q}$ are in $\mathcal{V}(\mathcal{Q})$, the rightward projection of a on $\partial\mathcal{Q}$ and the downward projection of b on

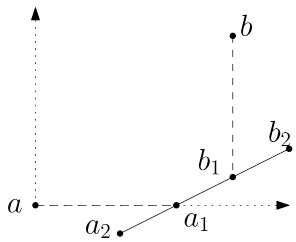

Figure 7: Illustrating the proof of Lemma 4: a_1 is the rightward projection of a on $\partial\mathcal{Q}_{core}$ and b_1 is the downward projection of b on $\partial\mathcal{Q}_{core}$.

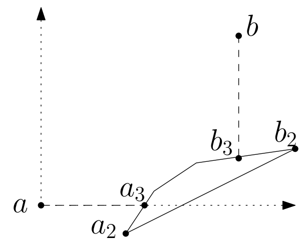

Figure 8: Illustrating the proof of Lemma 4: a_3 is the rightward projection of a on $\partial\mathcal{Q}$ and b_3 is the downward projection of b on $\partial\mathcal{Q}$. Both a_3 and b_3 must be on the same elementary curve β.

$\partial\mathcal{Q}$ must be both on β (e.g., see Fig. 8); we denote these two projection points by a_3 and b_3, respectively. By the construction of $G_{old}(\mathcal{M})$, there must be an xy-monotone path from a to b in $G_{old}(\mathcal{M})$ that is a concatenation of $\overline{aa_3}$, the portion of β between a_3 and b_3, and $\overline{b_3b}$ (note that a_3 is a type-1 Steiner point defined by a and b_3 is a type-1 Steiner point defined by b in $G_{old}(\mathcal{M})$).

Case 2 If no core of $\mathcal{Q}_{core}$ intersects the interior of the rectangle $R(a,b)$, then by the construction of G, there must be a cut-line l between a and b such that on l, a defines a Steiner point $a_h(l)$ and b defines a Steiner point $b_h(l)$ (e.g., see Fig. 9). Thus, there is an xy-monotone path from a to b in G consisting of $\overline{aa_h(l)} \cup \overline{a_h(l)b_h(l)} \cup \overline{b_h(l)b}$. Below, we show that there is also an xy-monotone path from a to b in $G_{old}(\mathcal{M})$.

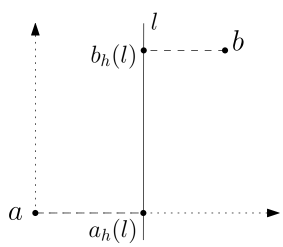

Figure 9: Illustrating the proof of Lemma 4: $a_h(l)$ is the rightward projection of a on l and $b_h(l)$ is the leftward projection of b on l.

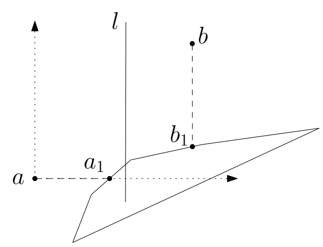

Figure 10: Illustrating the proof of Lemma 4: a_1 is the rightward projection of a on $\partial\mathcal{Q}$ and b_1 is the downward projection of b on $\partial\mathcal{Q}$. Both a_1 and b_1 must be on the same elementary curve β.

Since both G and $G_{old}(\mathcal{M})$ are built on $\mathcal{V}(\mathcal{Q})$, they have the same cut-line tree. Hence, the cut-line l still exists in $G_{old}(\mathcal{M})$. If both a and b are horizontally visible to l, then they still define Steiner points on l and consequently there is also an xy-monotone path from a to b in $G_{old}(\mathcal{M})$. Otherwise, we assume that a is not horizontally visible to l. Let a_1 be the rightward projection of a on $\partial\mathcal{Q}$ (see Fig. 10). Hence, a_1 must be between l and a. Let β be the elementary curve that contains a_1. Thus, β intersects the lower edge of $R(a,b)$ at a_1. Since $R(a,b)$ does not contain any point of $\mathcal{V}(\mathcal{Q}) - \{a,b\}$, the two endpoints of β are not in $R(a,b)$ and thus the downward projection of b on $\partial\mathcal{Q}$, denoted by b_1, must be on β as well. By the construction of $G_{old}(\mathcal{M})$, there must be an xy-monotone path from a to b in $G_{old}(\mathcal{M})$ that is the concatenation of $\overline{aa_1}$, the portion of β between a_1 and b_1, and $\overline{b_1b}$.

The above arguments prove that a shortest path from u to v in $G_{old}(\mathcal{M})$ corresponds to a shortest path from u to v in the plane.

It remains to show that a shortest u-v path in $G_{old}(\mathcal{M})$ corresponds to a shortest u-v path in $G_E(\mathcal{M})$. This can be seen easily since for any edge e in $G_{old}(\mathcal{M})$, if e is not in $G_E(\mathcal{M})$, then e is a path in $G_E(\mathcal{M})$. The lemma thus follows. □

The next lemma gives an algorithm for computing the graph $G_E(\mathcal{M})$.

Lemma 5. *The graph $G_E(\mathcal{M})$ can be computed in $O(n + h \log^{3/2} h 2^{\sqrt{\log h}})$ time.*

Proof. The algorithm for constructing $G_E(\mathcal{M})$ is similar to that for G_E in Lemma 1. We first triangulate the free space $\mathcal{F}$ in $O(n + h \log^{1+\epsilon} h)$ time for any constant $\epsilon > 0$ [2], after which computing the extended corridor structure takes $O(n + h \log h)$ time [8, 9, 11]. Subsequently, we obtain $\mathcal{Q}$ and the vertex set $\mathcal{V}(\mathcal{Q})$. All corridor paths are also available.

We compute the four projections of each point of $\mathcal{V}(\mathcal{Q})$ on $\partial\mathcal{Q}$ as type-1 Steiner points, which can be done after we compute the vertical and horizontal visibility decompositions of $\mathcal{Q}$ in $O(n + h \log^{1+\epsilon} h)$ time [2]. The graph edges for connecting each point of $\mathcal{V}(\mathcal{Q})$ to its four projection points on $\partial\mathcal{Q}$ can be obtained as well.

Next, we compute the type-2 and type-3 Steiner points. Since $|\mathcal{V}(\mathcal{Q})| = O(h)$, the cut-line tree $T(\mathcal{M})$ can be computed in $O(h \log h)$ time. Then, we determine the Steiner points on the cut-lines by traversing the tree $T(\mathcal{M})$ from top to bottom in a similar way as in Lemma 1. Since we have obtained the four projection points for each point of $\mathcal{V}(\mathcal{Q})$, computing all Steiner points on the cut-lines takes $O(h\sqrt{\log h}2^{\sqrt{\log h}})$ time. In the meanwhile, as in Lemma 1, for each point $p \in \mathcal{V}(\mathcal{Q})$, we compute the edges of $G_E(\mathcal{M})$ connecting the Steiner points defined by p on consecutive cut-lines in each super-level of $T(\mathcal{M})$ (i.e., the edges illustrated by the dotted segments in Fig 2). As in Lemma 1, all these graph edges can be computed in time linear in the total number of type-2 and type-3 Steiner points, which is $O(h\sqrt{\log h}2^{\sqrt{\log h}})$.

It remains to compute the graph edges of $G_E(\mathcal{M})$ connecting consecutive graph nodes on each elementary curve of $\mathcal{Q}$ and the graph edges connecting every two consecutive Steiner points (if they are visible to each other) on each cut-line.

On each connected component Q of $\mathcal{Q}$, we could compute a sorted list of all Steiner points and the points of $\mathcal{V}(Q)$ by sorting all these points and all obstacle vertices of Q along ∂Q. But that would take $O(n \log n)$ time in total because there are $O(n)$ obstacle vertices on all components of $\mathcal{Q}$. To do better, we take the following approach. For each elementary curve β, we sort all Steiner points on β by either their x-coordinates or y-coordinates. Since β is xy-monotone, such an order is also an order along β. Then, we merge the Steiner points thus ordered with the obstacle vertices on β, in linear time. Since there are $O(h)$ Steiner points on $\partial\mathcal{Q}$, it takes a total of $O(n + h \log h)$ time to sort the Steiner points and obstacle vertices on all elementary curves of $\mathcal{Q}$. After that, the edges of $G_E(\mathcal{M})$ on all elementary curves can be computed immediately.

We now compute the graph edges on the cut-lines connecting consecutive Steiner points. We first sort all Steiner points on each cut-line. This sorting step can be done in

$O(h \log^{3/2} h 2^{\sqrt{\log n}})$ time for all cut-lines. For each pair of consecutive Steiner points p and q on every cut-line, we determine whether p is visible to q by checking whether the upward projections of p and q on $\partial\mathcal{Q}$ are equal, and these upward projections can be performed in $O(\log n)$ time using the vertical visibility decomposition of $\mathcal{Q}$. Hence, the graph edges on all cut-lines are computed in $O(h\sqrt{\log h}2^{\sqrt{\log h}} \cdot \log n)$ time.

In summary, we can compute the graph $G_E(\mathcal{M})$ in $O(n+h\sqrt{\log h}2^{\sqrt{\log h}} \cdot \log n)$ time. Note that $h\sqrt{\log h}2^{\sqrt{\log h}} \cdot \log n = O(n + h \log^{3/2} h 2^{\sqrt{\log h}})$. To see this, if $h = O(\sqrt{n})$, then $n + h\sqrt{\log h}2^{\sqrt{\log h}} \cdot \log n = O(n)$; otherwise $\log n = \Theta(\log h)$. The lemma thus follows. $\square$

Consider any two query points s and t in the ocean $\mathcal{M}$. We define the gateway sets $V_g(s, G_E(\mathcal{M}))$ for s and $V_g(t, G_E(\mathcal{M}))$ for t on $G_E(\mathcal{M})$, as follows. We only discuss $V_g(s, G_E(\mathcal{M}))$; $V_g(t, G_E(\mathcal{M}))$ is similar. The definition of $V_g(s, G_E(\mathcal{M}))$ is analogous to that of $V_g(s, G_E)$. Specifically, $V_g(s, G_E(\mathcal{M}))$ has two subsets $V_g^1(s, G_E(\mathcal{M}))$ and $V_g^2(s, G_E(\mathcal{M}))$. $V_g^2(s, G_E(\mathcal{M}))$ is defined in the same way as $V_g^2(s, G_E)$, and thus $|V_g^2(s, G_E(\mathcal{M}))| = O(\sqrt{\log h})$. $V_g^1(s, G_E(\mathcal{M}))$ is defined with respect to the elementary curves of $\mathcal{Q}$, as follows. Let q be the rightward projection point of s on $\partial\mathcal{Q}$. Suppose q is on the elementary curve β and p_1 and p_2 are the two nodes of $G_E(\mathcal{M})$ on β adjacent to q. Then p_1 and p_2 are in $V_g^1(s, G_E(\mathcal{M}))$, and for each $p \in \{p_1, p_2\}$, we define a gateway edge from s to p consisting of $\overline{sq}$ and the portion of β between q and p. Similarly, for each of the leftward, upward, and downward projections of s on $\partial\mathcal{Q}$, there are at most two gateways in $V_g^1(s, G_E(\mathcal{M}))$.

The next lemma shows that the gateways of $V_g(s, G_E(\mathcal{M}))$ "control" the shortest paths from s to all points of $\mathcal{V}(\mathcal{Q})$.

Lemma 6. *For any point p of $\mathcal{V}(\mathcal{Q})$, there exists a shortest path from s to p using $G_E(\mathcal{M})$ that contains a gateway of s in $V_g(s, G_E(\mathcal{M}))$.*

Proof. We define a gateway set $V_g(s, G_{old}(\mathcal{M}))$ for s on the graph $G_{old}(\mathcal{M})$, as follows. The set $V_g(s, G_{old}(\mathcal{M}))$ has two subsets $V_g^1(s, G_{old}(\mathcal{M}))$ and $V_g^2(s, G_{old}(\mathcal{M}))$. The first subset $V_g^1(s, G_{old}(\mathcal{M}))$ is exactly the same as $V_g^1(s, G_E(\mathcal{M}))$, and the second subset $V_g^2(s, G_{old}(\mathcal{M}))$ contains gateways on the cut-lines of $T(\mathcal{M})$, which are defined similarly as $V_g^2(s, G_{old})$ on G_{old} and $T(\mathcal{P})$, discussed in Section 2. Note that the gateways in $V_g(s, G_{old}(\mathcal{M}))$ are exactly those nodes of $G_{old}(\mathcal{M})$ that are adjacent to s if we "insert" s into the graph $G_{old}(\mathcal{M})$ (similar arguments were used for $V_g(s, G_{old})$ in [7]). Hence, there exists a shortest path from s to p using $G_{old}(\mathcal{M})$ that contains a gateway of s in $V_g(s, G_{old}(\mathcal{M}))$.

Since the graph $G_E(\mathcal{M})$ is defined analogously as G_E and $G_{old}(\mathcal{M})$ is defined analogously as G_{old}, by using a similar analysis as in the proof of Lemma 2, we can show that there exists a shortest path from s to p using $G_E(\mathcal{M})$ that contains a gateway of s in $V_g(s, G_E(\mathcal{M}))$. We omit the details. The lemma thus follows. $\square$

Similar results also hold for the gateway set $V_g(t, G_E(\mathcal{M}))$ of t. We have the following corollary.

Corollary 2. *If there exists a shortest s-t path through a point of $\mathcal{V}(\mathcal{Q})$, then there exists a shortest s-t path through a gateway of s in $V_g(s, G_E(\mathcal{M}))$ and a gateway of t in $V_g(t, G_E(\mathcal{M}))$.*

The following lemma gives an algorithm for computing the gateways.

Lemma 7. *With a preprocessing of $O(n+h\cdot\log^{3/2} h\cdot 2^{\sqrt{\log h}})$ time and $O(n+h\cdot\sqrt{\log h}\cdot 2^{\sqrt{\log h}})$ space, the gateway sets $V_g(s, G_E(\mathcal{M}))$ and $V_g(t, G_E(\mathcal{M}))$ can be computed in $O(\log n)$ time for any two query points s and t in $\mathcal{M}$.*

Proof. The algorithm is similar to that for Lemma 3; we only point out the differences. We discuss our algorithm only for computing $V_g(s, G_E(\mathcal{M}))$; the case for $V_g(t, G_E(\mathcal{M}))$ is similar.

To compute $V_g^1(s, G_E(\mathcal{M}))$, we build the horizontal and vertical visibility decompositions of $\mathcal{Q}$. Then, the four projections of s on $\partial\mathcal{Q}$ can be determined in $O(\log n)$ time. Consider any such projection p of s. Suppose p is on an elementary curve β. We need to determine the two nodes of $G_E(\mathcal{M})$ on β adjacent to p, which are gateways of $V_g^1(s, G_E(\mathcal{M}))$. We maintain a sorted list of all nodes of $G_E(\mathcal{M})$ on β, and do binary search to find these two gateways of s on β in this sorted list by using only the y-coordinates (or the x-coordinates) of the nodes since β is xy-monotone. Also, since β is xy-monotone, for any two points q and q' on β, the length of the portion of β between q and q' is equal to the length of $\overline{qq'}$. Hence, after these two gateways of s on β are found, the lengths of the two gateway edges from s to them can be computed in constant time. Since $V_g^1(s, G_E(\mathcal{M}))$ has $O(1)$ gateways, $V_g^1(s, G_E(\mathcal{M}))$ can be computed in $O(\log n)$ time.

To compute $V_g^2(s, G_E(\mathcal{M}))$, we take the same approach as for Lemma 3. In the preprocessing, for every cut-line l, we maintain a sorted list of all Steiner points on l, and associate with each such Steiner point its upward and downward projections on $\partial\mathcal{Q}$. Computing these projections for each Steiner point takes $O(\log n)$ time. Then we build a fractional cascading data structure [5] for the sorted lists of Steiner points on all cut-lines along the cut-line tree $T(\mathcal{M})$. Using this fractional cascading data structure, the gateway set $V_g^2(s, G_E(\mathcal{M}))$ can be computed in $O(\log h)$ time. Note that the time is $O(\log h)$ (instead of $O(\log n)$) since the size of $G_E(\mathcal{M})$ is $O(h\sqrt{\log h}2^{\sqrt{\log h}})$.

The preprocessing takes $O(n + h\sqrt{\log h}2^{\sqrt{\log h}}\log n)$ time and $O(n + h\sqrt{\log h}2^{\sqrt{\log h}})$ space. Note that $n + h\sqrt{\log h}2^{\sqrt{\log h}}\log n = O(n + h\log^{3/2} h2^{\sqrt{\log h}})$. The lemma thus follows. □

We summarize our algorithm in Lemma 8 for the case when both query points are in $\mathcal{M}$.

Lemma 8. *With a preprocessing of $O(n + h^2\log^2 h4^{\sqrt{\log h}})$ time and $O(n + h^2\log h4^{\sqrt{\log h}})$ space, each two-point query can be answered in $O(\log n)$ time for any two query points in the ocean $\mathcal{M}$.*

Proof. In the preprocessing, we build the graph $G_E(\mathcal{M})$, and for each node v of $G_E(\mathcal{M})$, compute a shortest path tree in $G_E(\mathcal{M})$ from v. We maintain a shortest path length table such that for any two nodes u and v in $G_E(\mathcal{M})$, the shortest path length between u and v can be found in $O(1)$ time. Since $G_E(\mathcal{M})$ has $O(h\sqrt{\log h}2^{\sqrt{\log h}})$ nodes and edges, computing and maintaining all shortest path trees in $G_E(\mathcal{M})$ take $O(h^2 \log h 4^{\sqrt{\log h}})$ space and $O(n + h^2 \log^2 h 4^{\sqrt{\log h}})$ time.

To report an actual shortest path in the plane in time linear in the number of edges of the output path, we need to maintain additional information. Consider an elementary curve β of $\mathcal{Q}$. Let u and v be two consecutive nodes of $G_E(\mathcal{M})$ on β. By our definition of $G_E(\mathcal{M})$, there is an edge (u, v) in $G_E(\mathcal{M})$. If the edge (u, v) is contained in our output path, we need to report all obstacle vertices and edges of β between u and v. For this, on each elementary curve β, we explicitly maintain a list of obstacle edge between each pair of consecutive nodes of $G_E(\mathcal{M})$ along β. Since the total number of nodes of $G_E(\mathcal{M})$ on all elementary curves is $O(h)$ and the total number of obstacle vertices of $\mathcal{Q}$ is $O(n)$, maintaining such *edge lists* for all elementary curves takes $O(n)$ space.

In addition, we also perform the preprocessing for Lemma 7.

The overall preprocessing takes $O(n+h^2 \log^2 h 4^{\sqrt{\log h}})$ time and $O(n+h^2 \log h 4^{\sqrt{\log h}})$ space.

Now consider any two query points s and t in $\mathcal{M}$. As for Theorem 1, we first check whether there exists a trivial shortest s-t path. But trivial shortest paths here are defined with respect to the elementary curves of $\mathcal{Q}$ instead of the obstacle edges of $\mathcal{P}$. For example, consider s^r (i.e., the rightward projection of s on $\partial\mathcal{Q}$) and t^d. If $\overline{ss^r}$ intersects $\overline{tt^d}$, then there is a trivial shortest s-t path $\overline{sq} \cup \overline{qt}$, where $q = \overline{ss^r} \cap \overline{tt^d}$; otherwise, if s^r and t^d are both on the same elementary curve β of $\mathcal{Q}$, then there is a trivial shortest s-t path which is the concatenation of $\overline{ss^r}$, the portion of β between s^r and t^d, and $\overline{t^d t}$. Similarly, trivial shortest s-t paths are also defined by other projections of s and t on $\partial\mathcal{Q}$.

We can determine whether there exists a trivial shortest s-t path in $O(\log n)$ time by using the vertical and horizontal decompositions of $\mathcal{Q}$ to compute the four projection points of s and t on $\partial\mathcal{Q}$. If yes, we find such a shortest path in additional time linear in the number of edges of the output path. Note that for the case, e.g., when s^r and t^d are both on the same elementary curve β, the output path may not be of $O(1)$ size since there may be multiple obstacle vertices on the portion of β between s^r and t^d; but we can still output such a path in linear time by using the edge lists we maintain on each elementary curve. Below, we assume there is no trivial shortest s-t path.

By using the cores of $\mathcal{Q}$ in the proof of Lemma 4 and a similar analysis as in [7], we can show that there must be a shortest s-t path that contains at least one point of $\mathcal{V}(\mathcal{Q})$. By Corollary 2, there exists a shortest s-t path through a gateway of s and a gateway of t in $G_E(\mathcal{M})$. Using Lemma 7, we compute the two gateway sets $V_g(s, G_E(\mathcal{M}))$ and $V_g(t, G_E(\mathcal{M}))$. By building a gateway graph for s and t as in Theorem 1, we can compute the length of a shortest s-t path in $O(\log h)$ time since $|V_g(s, G_E(\mathcal{M}))| = O(\sqrt{\log h})$, $|V_g(t, G_E(\mathcal{M}))| = O(\sqrt{\log h})$, and thus the gateway graph has $O(\sqrt{\log h})$ nodes and $O(\log h)$ edges. An actual path can then be reported in additional time linear in the number of edges of the output path, by using the shortest path trees of $G_E(\mathcal{M})$ and the edge lists maintained on the elementary curves, as discussed above. The lemma thus follows. $\square$

4.3 The General Queries

In this section, we show how to handle the general queries in which at least one query point is not in $\mathcal{M}$. We first consider the case where neither s nor t is in a canal. The canal case where at least one of the query points is in a canal will be discussed in Section 4.3.4, where the case is handled by similar techniques although it is a little more complicated since each canal has two gates.

Without loss of generality, we assume that s is in a bay, denoted by B. The point t can be in B, $\mathcal{M}$, or another bay, and we discuss these three cases in the following three subsections.

Let g denote the gate of B. We will discuss several possible cases that a shortest s-t path may cross the gate g. For each case, we will compute some "candidate" paths and select the one with the smallest length as our solution.

4.3.1 The Query Point t is in B

When the query point t is in B, we have the following lemma.

Lemma 9. *If B is a bay and $t \in B$, then there exists a shortest s-t path in B.*

Proof. Let π be any shortest s-t path in the plane. If π is in B, then we are done. Otherwise, π must intersect the only gate g of B; further, since both s and t are in B, if π exits from B (through g), then it must enter B again (through g as well). Let p be the first point on g encountered as going from s to t along π and let q be the last such point on g. Let π' be the s-t path obtained by replacing the portion of π between p and q by $\overline{pq} \subseteq g$. Note that π' is in B. Since $\overline{pq}$ is a shortest path from p to q, π' is also a shortest s-t path. The lemma thus follows. $\square$

To handle the case of $t \in B$, in the preprocessing, we build a data structure for two-point Euclidean shortest path queries in B, denoted by $\mathcal{D}(B)$, in $O(|B|)$ time and space [20]. Since a Euclidean shortest path in any simple polygon is also an L_1 shortest path [24] and B is a simple polygon, for $t \in B$, we can use $\mathcal{D}(B)$ to answer the shortest s-t path query in B in $O(\log n)$ time.

4.3.2 The Query Point t is in $\mathcal{M}$

If the query point t is in $\mathcal{M}$, then a shortest s-t path must cross the gate g of B. A main difficulty for answering the general queries is to deal with this case. More specifically, we already have a graph $G_E(\mathcal{M})$ on $\mathcal{M}$, and our goal is to design a mechanism to connect the bay B with $G_E(\mathcal{M})$ through the gate g, so that it can capture the shortest path information in the union of B and $\mathcal{M}'$ (recall that $\mathcal{M}'$ is the union of $\mathcal{M}$ and all corridor paths).

We begin with some observations on how a shortest s-t path may cross g. Without loss of generality, we assume that g has a positive slope and the interior of B on g is above g. Let a_1 and a_2 be the two endpoints of g such that a_1 is higher than a_2 (see Fig. 11). Let

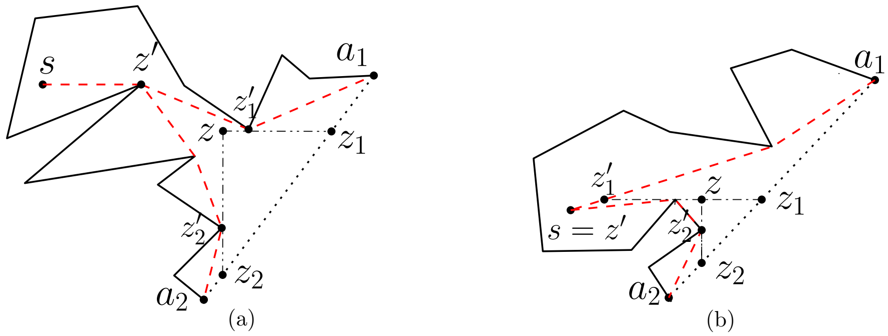

Figure 11: Illustrating the definitions of z', z'_1, z'_2, z, z_1, and z_2. In (a), $\overline{z_1 z'_1}$ is tangent to $\pi(z', a_1)$ (at z'_1); in (b), $\overline{z_1 z'_1}$ is tangent to $\pi(z', a_2)$.

$\pi(s, a_1)$ (resp., $\pi(s, a_2)$) be the Euclidean shortest path in B from s to a_1 (resp., a_2). Let z' be the farthest point from s on $\pi(s, a_1) \cap \pi(s, a_2)$ (possibly $z' = s$). Let $\pi(z', a_1)$ (resp., $\pi(z', a_2)$) be the subpath of $\pi(s, a_1)$ (resp., $\pi(s, a_2)$) between z' and a_1 (resp., a_2). It is well known that both $\pi(z', a_1)$ and $\pi(z', a_2)$ are convex chains [21, 34], and the region enclosed by $\pi(z', a_1)$, $\pi(z', a_2)$, and g in B is a "funnel" with z' as the *apex* and g as the *base* [21, 34] (see Fig. 11). Let F denote this funnel and ∂F denote its boundary.

We define four special points z'_1, z'_2, z_1, and z_2 (see Fig. 11). Suppose we move along $\pi(z', a_1)$ from z'; let z'_1 be the first point on $\pi(z', a_1)$ we encounter that is *horizontally* visible to $g = \overline{a_1 a_2}$. Similarly, as moving along $\pi(z', a_2)$ from z', let z'_2 be the first point on $\pi(z', a_2)$ encountered that is *vertically* visible to g. Note that in some cases z'_1 (resp., z'_2) can be z', a_1, or a_2. Let z_1 be the horizontal projection of z'_1 on g and z_2 be the vertical projection of z'_2 on g (see Fig. 11).

The points z_1 and z_2 are particularly useful. We first have the following observation.

Observation 1. *The point z_1 is above z_2, i.e., the y-coordinate of z_1 is no smaller than that of z_2.*

Proof. If z' is either a_1 or a_2, then by their definitions, we have $z_1 = z_2 = z'_1 = z'_2 = z'$ and the observation trivially holds. Suppose z' is neither a_1 nor a_2. If $z_1 = a_1$, then the observation also holds since a_1 is the highest point on g. We assume $z_1 \neq a_1$, which implies $z'_1 \neq a_1$.

Let $\pi(z'_1, a_1)$ be the portion of $\pi(z', a_1)$ between z'_1 and a_1. Note that the "pseudo-triangular" region enclosed by $\overline{a_1 z_1}$, $\overline{z_1 z'_1}$, and $\pi(z'_1, a_1)$ does not contain any point of ∂B in its interior. For any point p in the interior of $\overline{a_1 z_1}$, since $\pi(z'_1, a_1)$ is convex and $\overline{z_1 z'_1}$ is horizontal, p must be vertically visible to $\pi(z'_1, a_1)$, say, at a point $q \in \pi(z'_1, a_1)$. Clearly, q is not z'_1. Hence, the line containing $\overline{pq}$ cannot be tangent to $\pi(z', a_1)$ at q, implying that q is not z'_2. Therefore, the point p must be strictly above z_2. Since p is an arbitrary point in the interior of $\overline{a_1 z_1}$, z_1 must be above z_2. The observation thus follows. □

Lemma 10. *For any point $p \in \overline{a_1 z_1}$, there is a shortest path from s to p that contains z_1; likewise, for any point $p \in \overline{a_2 z_2}$, there is a shortest path from s to p that contains z_2.*

Proof. We only prove the case of $p \in \overline{a_1 z_1}$ since the other case of $p \in \overline{a_2 z_2}$ is symmetric. It suffices to show that there exists a shortest path from z' to $p \in \overline{a_1 z_1}$ that contains z_1.

Recall that z_1 is the horizontal projection of z_1' on g. Let $\pi(z_1', a_1)$ be the portion of $\pi(z', a_1)$ between z_1' and a_1. Consider the "pseudo-triangular" region R enclosed by $\overline{a_1 z_1}$, $\overline{z_1 z_1'}$, and $\pi(z_1', a_1)$. Since $\pi(z_1', a_1)$ is convex, every point on $\pi(z_1', a_1)$ is horizontally visible to g.

We claim that there exists a shortest path π from z' to p that intersects $\overline{z_1' z_1}$. Indeed, if $z' = z_1'$, then the claim is trivially true. Otherwise, since z_1' is the first point on $\pi(z', a_1)$ that is horizontally visible to g if we go from z' to a_1 along $\pi(z', a_1)$, z' cannot be horizontally visible to g, and thus, z' is not in R. Note that $\overline{z_1 z_1'}$ partitions the funnel F into two parts, one of which is R. Also, the funnel F contains a shortest path π from z' to p. Since $p \in R$ and $z' \notin R$, the path π must intersect $\overline{z_1' z_1}$. The claim is proved.

Suppose π intersects $\overline{z_1' z_1}$ at a point q. Since $\overline{q z_1} \cup \overline{z_1 p}$ is xy-monotone (and thus is a shortest path), we can obtain another shortest path from z' to p that contains z_1 by replacing the portion of π between q and p by $\overline{q z_1} \cup \overline{z_1 p}$. The lemma thus follows. □

For the case of $t \in \mathcal{M}$, Lemma 10 implies the following. Let $i \in \{1, 2\}$. If a shortest s-t path crosses g at a point on $\overline{a_i z_i}$, then there must be a shortest s-t path that is a concatenation of a shortest path $\pi(s, z_i)$ from s to z_i in B and a shortest path $\pi(z_i, t)$ from z_i to t in $\mathcal{M}'$. The path $\pi(s, z_i)$ can be found using the data structure $\mathcal{D}(B)$ and $\pi(z_i, t)$ can be found by Lemma 8 since both z_i and t are in $\mathcal{M}$. Hence, such a shortest s-t path query can be answered in $O(\log n)$ time, provided that we can find z_i in $O(\log n)$ time (as to be shown in Lemma 17).

It remains to consider the case where every shortest s-t path crosses the interior of $\overline{z_1 z_2}$; in other words, no shortest s-t paths cross $\overline{a_1 z_1} \cup \overline{a_2 z_2}$.

Let z denote the intersection of the horizontal line containing $\overline{z_1 z_1'}$ and the vertical line containing $\overline{z_2 z_2'}$ (see Fig. 11). The point z is useful as shown by the next lemma.

Lemma 11. *The point z is in the funnel F, and for any point $p \in \overline{z_1 z_2}$, there is a shortest path from s to p that contains z.*

Proof. We first prove $z \in F$. For this, it suffices to prove that the interior of the triangle $\triangle z z_1 z_2$ does not contain any point on the boundary of F. Let R denote the interior of $\triangle z z_1 z_2$.

Assume to the contrary that R intersects ∂F. Let q be any point in $R \cap \partial F$ that is horizontally visible to $\overline{z_1 z_2}$. Such a point q always exists if $R \cap \partial F \neq \emptyset$. Note that q is on either $\pi(z', a_1)$ or $\pi(z', a_2)$. Without loss of generality, assume q is on $\pi(z', a_1)$. Observe that $\pi(z_1', a_1)$ is xy-monotone since z_1' is horizontally visible to g. Because q is also horizontally visible to g, by the definition of z_1', q must be on $\pi(z_1', a_1)$. Since q is in R, q must be strictly below z_1'. Since a_1 is no lower than z_1, a_1 is also no lower than z_1'. Thus, when following the path $\pi(z_1', a_1)$ from z_1' to a_1, we have to strictly go down (through q) and then go up (to a_1), which contradicts with that the fact the path $\pi(z_1', a_1)$ is xy-monotone. Hence, R cannot contain any point on ∂F and z must be in F.

Consider any point $p \in \overline{z_1 z_2}$. Below we prove that there is a shortest path from s to p containing z. It suffices to show that there exists a shortest path from z' to p containing z. If $z_1 = z_2$, then $z = z_1 = z_2 = p$ and we are done. Below we assume $z_1 \neq z_2$, which implies $z_1 \neq a_2$ since otherwise $z_1 = z_2$ by Observation 1; similarly, $z_2 \neq a_1$. Note that $z_1 \neq z_2$ also implies $z' \notin \{a_1, a_2\}$.

Let $\pi(z', p)$ be a shortest path in F from z' to p. Let l_1 be the horizontal line containing $\overline{z_1 z_1'}$ and l_2 be the vertical line containing $\overline{z_2 z_2'}$.

In the following, we first prove that $l_1 \cap F$ is a line segment and it must intersect the path $\pi(z', p)$. Consider the line segment $\overline{z_1 z_1'}$. Depending on whether l_1 is tangent to $\pi(z', a_1)$ at z_1', there are two possible cases (e.g., see Fig. 11).

1. If l_1 is tangent to $\pi(z', a_1)$ at z_1' (see Fig. 11(a)), then we extend $\overline{z_1 z_1'}$ horizontally leftwards until it hits ∂F, say, at a point z_1''. Since $\pi(z', a_1)$ is convex, z' is above the line l_1 and z_1'' is on $\pi(z', a_2)$. Since $\pi(z', a_2)$ is also convex and z' is above l_1, we obtain $l_1 \cap F = \overline{z_1 z_1''}$.

   Observe that $\overline{z_1' z_1''}$ partitions F into two sub-polygons such that z' and p are in different sub-polygons. Hence, the path $\pi(z', p)$ must intersect $\overline{z_1' z_1''} \subseteq l_1 \cap F$, which is a line segment.

2. If l_1 is not tangent to $\pi(z', a_1)$ at z_1', then depending on whether $z_1' = z'$, there are two subcases.

   (a) If $z_1' = z'$, then due to the convexity of $\pi(z', a_1)$ and $\pi(z', a_2)$, we have $l_1 \cap F = \overline{z_1 z_1'}$. Since $z' = z_1'$, it is trivially true that $\pi(z', p)$ intersects $l_1 \cap F = \overline{z_1 z_1'}$.

   (b) If $z_1' \neq z'$ (see Fig. 11(b)), then we claim that $\overline{z_1 z_1'}$ must be tangent to $\pi(z', a_2)$ at a point, say, z_1''. Suppose to the contrary that this is not the case. Then, since $z_1' \neq z'$, $z_1 \neq a_2$, and l_1 is not tangent to $\pi(z', a_1)$ at z_1', we can move l_1 downwards by an infinitesimal value such that the new l_1 intersects g at a point z_3 and intersects $\pi(z', a_1)$ at a point z_3' such that z_3' is horizontally visible to z_3. Clearly, z_3' is on $\pi(z', a_1)$ between z' and z_1'. But this contradicts with the definition of z_1', i.e., z_1' is the first point on $\pi(z', a_1)$ horizontally visible to g if we go from z' to a_1 along $\pi(z', a_1)$. The claim is thus proved.

   By the above claim and the convexity of $\pi(z', a_2)$, z' is below l_1. Also by the convexity of $\pi(z', a_1)$, we have $l_1 \cap F = \overline{z_1 z_1'}$. Further, observe that $\overline{z_1' z_1''}$ partitions F into two sub-polygons such that z' and p are in different sub-polygons. Hence, the path $\pi(z', p)$ must intersect $\overline{z_1' z_1''} \subseteq l_1 \cap F$.

   Therefore, $l_1 \cap F$ is a line segment that intersects $\pi(z', p)$.

The above arguments prove that $l_1 \cap F$ is a line segment that intersects the path $\pi(z', p)$, say, at a point q_1. By using a similar analysis, we can also show that $l_2 \cap F$ is a line segment that intersects $\pi(z', p)$, say, at a point q_2. Note that this implies that z is on the intersection of the segment $l_1 \cap F$ and the segment $l_2 \cap F$. Since $\overline{q_1 z} \cup \overline{z q_2}$ is xy-monotone (and thus is a shortest path), if we replace the subpath of $\pi(z', p)$ between q_1 and q_2 by $\overline{q_1 z} \cup \overline{z q_2}$ to obtain another path $\pi'(z', p)$ from z' to p, then $\pi'(z', p)$ is still a shortest path. Since $\pi'(z', p)$ contains z, the lemma follows. □

If there is a shortest s-t path crossing g at a point on $\overline{z_1 z_2}$, then by Lemma 11, there is a shortest s-t path that is a concatenation of a shortest path from s to z in B and a shortest path from z to t (which crosses g). A shortest s-z path in B can be found by using the data structure $\mathcal{D}(B)$ in $O(\log n)$ time, provided that we can compute z in $O(\log n)$ time. It remains to compute a shortest z-t path that crosses g at a point on $\overline{z_1 z_2}$. Note that such a shortest z-t path may or may not contain a point in $\mathcal{V}(g) \cap \overline{z_1 z_2}$, where $\mathcal{V}(g)$ is the set of points of $\mathcal{V}(\mathcal{Q})$ lying on g ($\mathcal{V}(g) = \emptyset$ is possible). We first discuss the former case and the latter case will be handled later in Lemma 16.

For the former case, we will build a graph $G_E(g)$ inside B and merge it with the graph $G_E(\mathcal{M})$ on $\mathcal{M}$ so that the merged graph allows to find a shortest z-t path crossing a point in $\mathcal{V}(g) \cap \overline{z_1 z_2}$. In the sequel, we introduce the graph $G_E(g)$. Let $h_g = |\mathcal{V}(g)|$.

The graph $G_E(g)$ is defined on the points of $\mathcal{V}(g)$ in a similar manner as G_E in Section 3. One difference is that $G_E(g)$ is built inside B and uses vertical *cut-segments* in B instead of cut-lines. Also, no type-1 Steiner point is needed for $G_E(g)$. Specifically, we define a *cut-segment tree* $T(g)$ as follows. The root u of $T(g)$ is associated with a point set $V(u) = \mathcal{V}(g)$. Each node u of $T(g)$ is also associated with a vertical cut-segment $l(u)$, defined as follows. Let p be the point of $V(u)$ with the median x-coordinate. Note that p is on g. We extend a vertical line segment from p upwards into the interior of B until it hits ∂B; this segment is the cut-segment $l(u)$. The left (resp., right) child of u is defined recursively on the points of $V(u)$ to the left (resp., right) of $l(u)$.

Clearly, $T(g)$ has $O(\log h_g)$ levels and $O(\sqrt{\log h_g})$ super-levels. We define the type-2 and type-3 Steiner points on the cut-segments of $T(g)$ in the same way as in Section 3. The graph $G_E(g)$ is then defined similarly as G_E in Section 3. We omit the details. $G_E(g)$ has $O(h_g\sqrt{\log h_g}2^{\sqrt{\log h_g}})$ nodes and $O(h_g\sqrt{\log h_g}2^{\sqrt{\log h_g}})$ edges.

Let n_B denote the number of obstacle vertices of the bay B.

Lemma 12. *The graph $G_E(g)$ can be constructed in $O(n_B + h_g \cdot \log^{3/2} h_g \cdot 2^{\sqrt{\log h_g}})$ time.*

Proof. To compute the cut-segments of $T(g)$, for each point $p \in \mathcal{V}(g)$, we need to compute the first point on the boundary of B hit by extending a vertical line segment from p upwards. For this, we first compute the vertically visible region of B from the segment g using the linear time algorithms in [29, 33], and then find all such cut-segments from the points of $\mathcal{V}(g)$, in $O(n_B + h_g)$ time. The cut-segment tree $T(g)$ can then be computed in $O(h_g \log h_g)$ time.

To compute the Steiner points on the cut-segments, for each point $p \in \mathcal{V}(g)$, we find the first point $p_h(B)$ on the boundary of B horizontally visible from p. The points $p_h(B)$ for all $p \in \mathcal{V}(g)$ can be computed in totally $O(n_B + h_g)$ time by using the algorithms in [29, 33].

Next, we compute the Steiner points on the cut-segments of $T(g)$. Determining whether a point $p \in \mathcal{V}(g)$ is horizontally visible to a cut-segment l (and if yes, put a corresponding Steiner point on l) takes $O(1)$ time using $p_h(B)$, as follows. We first check whether the y-coordinate of p is between the y-coordinate of the lower endpoint of l and that of the upper endpoint of l; if yes, we check whether l is between p and $p_h(B)$ (if yes, then p is horizontally visible to l); otherwise, p is not horizontally visible to l. Thus, all

Steiner points can be obtained in $O(h_g\sqrt{\log h_g}2^{\sqrt{\log h_g}})$ time.

For each cut-segment l, to compute the edges between consecutive graph nodes on l, it suffices to sort all Steiner points on l. The sorting on all cut-segments takes $O(h_g \cdot \log^{3/2} h_g \cdot 2^{\sqrt{\log h_g}})$ time. Hence, the total time for building the graph $G_E(g)$ is $O(n_B + h_g \cdot \log^{3/2} h_g \cdot 2^{\sqrt{\log h_g}})$. □

We define a gateway set $V_g(z, G_E(g))$ for z on $G_E(g)$ such that for any point $p \in \mathcal{V}(g) \cap \overline{z_1z_2}$, there is a shortest path from z to p using $G_E(g)$ containing a gateway of z. $V_g(z, G_E(g))$ is defined similarly as $V_g^2(s, G_E)$ in Section 3, but only on the Steiner points in the triangle $\triangle zz_1z_2$ (because $\triangle zz_1z_2$ contains a shortest path from z to any point in $\mathcal{V}(g) \cap \overline{z_1z_2}$). Specifically, for each *relevant projection cut-segment* l (defined similarly as the relevant projection cut-lines in Section 3) of z to the right of z, if z is horizontally visible to l, then the node of $G_E(g)$ on l immediately below the horizontal projection point of z on l is in $V_g(z, G_E(g))$. Thus, $|V_g(z, G_E(g))| = O(\sqrt{\log h_g})$.

Lemma 13. *For any point $p \in \mathcal{V}(g) \cap \overline{z_1z_2}$, there is a shortest path from z to p in B using $G_E(g)$ that contains a gateway of z in $V_g(z, G_E(g))$.*

Proof. Consider a point $p \in \mathcal{V}(g) \cap \overline{z_1z_2}$. Note that p defines a node in $G_E(g)$. Let l_p be the cut-segment through p. Since the triangle $\triangle zz_1z_2 \subseteq B$ and $p \in \overline{z_1z_2}$, z is horizontally visible to l_p.

If there is no other cut-segment of $T(g)$ strictly between z and l_p, then l_p must be a relevant projection cut-segment of z. Let p' be the gateway of z on l_p, i.e., the graph node on l_p immediately below the horizontal projection $z_h(l_p)$ of z on l_p. Note that the path $\overline{zz_h(l_p)} \cup \overline{z_h(l_p)p}$ is a shortest path from z to p since it is xy-monotone. Clearly, this path contains the gateway p'.

If there is at least one cut-segment strictly between z and l_p, then if l_p is a relevant cut-segment of z, we can prove the lemma by a similar analysis as above; otherwise, there is at least one node u in $T(g)$ such that $l(u)$ is a relevant projection cut-segment of z between z and p and p defines a Steiner point on $l(u)$ (this can be seen from the definition of the graph $G_E(g)$; we omit the details). Let $z_h(l(u))$ be the horizontal projection of z on $l(u)$ and $p_h(l(u))$ be the horizontal projection of p on $l(u)$. The path $\overline{zz_h(l(u))} \cup \overline{z_h(l(u))p_h(l(u))} \cup \overline{p_h(l(u))p}$ is a shortest path from z to p since it is xy-monotone. Because $p_h(l(u))$ is a Steiner point on $l(u)$, this path must contain a gateway of z on $l(u)$ (this gateway must be on $\overline{z_h(l(u))p_h(l(u))}$). The lemma thus follows. □

Since $\mathcal{V}(g) \subseteq \mathcal{V}(\mathcal{Q})$, each point of $\mathcal{V}(g)$ is also a node of $G_E(\mathcal{M})$. We merge the two graphs $G_E(\mathcal{M})$ and $G_E(g)$ into one graph, denoted by $G_E(\mathcal{M}, g)$, by treating the two nodes in these two graphs defined by the same point in $\mathcal{V}(g)$ as a single node. By Lemmas 6 and 13, we have the following result.

Lemma 14. *If a shortest s-t path contains a point in $\mathcal{V}(g) \cap \overline{z_1z_2}$, then there is a shortest s-t path along $G_E(\mathcal{M}, g)$ containing a gateway of z in $V_g(z, G_E(g))$ and a gateway of t in $V_g(t, G_E(\mathcal{M}))$.*

Proof. Let p be a point of $\mathcal{V}(g) \cap \overline{z_1 z_2}$ that is contained in a shortest s-t path. By Lemma 11, there is a shortest path from s to p that contains z. By Lemma 13, there is a shortest path from z to p that contains a gateway of z in $V_g(z, G_E(g))$. On the other hand, since both t and p are in the ocean $\mathcal{M}$ and $p \in \mathcal{V}(g) \subseteq \mathcal{V}(\mathcal{Q})$, by Lemma 6, there exists a shortest path from t to p that contains a gateway of t in $V_g(t, G_E(\mathcal{M}))$. This proves the lemma. □

By Lemma 14, if there is a shortest path from z to t that contains a point of $\mathcal{V}(g) \cap \overline{z_1 z_2}$, then we can use the gateways of both z and t to find a shortest path along the graph $G_E(\mathcal{M}, g)$. By using a similar algorithm as that for Lemma 3, we can compute the gateways of z on $G_E(g)$.

Lemma 15. *With a preprocessing of $O(h_g \log^{3/2} h_g 2^{\sqrt{\log h_g}})$ time and $O(h_g \sqrt{\log h_g} 2^{\sqrt{\log h_g}})$ space, we can compute the gateway set $V_g(z, G_E(g))$ of z in $O(\log h)$ time.*

Proof. The algorithm is similar to that in Lemma 3 for computing $V_g^2(s, G_E)$. One main difference is that here every two graph nodes on any cut-segment of $T(g)$ are visible to each other. As the preprocessing, we build a sorted list of the graph nodes on each cut-segment of $T(g)$, and construct a fractional cascading data structure [5] along $T(g)$ for the sorted lists of all cut-segments. Then for a point z, $V_g(z, G_E(g))$ can be computed in $O(\log h)$ time. □

So far, we have shown how to find a shortest s-t path if such a path contains a point in $\{z_1, z_2\} \cup \{\mathcal{V}(g) \cap \overline{z_1 z_2}\}$. It remains to handle the case when no shortest s-t path contains a point in $\{z_1, z_2\} \cup \{\mathcal{V}(g) \cap \overline{z_1 z_2}\}$ (including the case of $\mathcal{V}(g) = \emptyset$), i.e., no shortest path from z to t contains a point in $\{z_1, z_2\} \cup \{\mathcal{V}(g) \cap \overline{z_1 z_2}\}$. Lemma 16 below shows that in this case, $t \in \mathcal{M}$ must be horizontally visible to $\overline{z z_2}$ and thus there is a trivial shortest path from z to t.

Lemma 16. *If no shortest path $\pi(z, t)$ contains a point in $\mathcal{V}(g) \cap \overline{z_1 z_2}$ (this includes the case of $\mathcal{V}(g) = \emptyset$), then t must be horizontally visible to $\overline{z z_2}$.*

Proof. Let the points of $\mathcal{V}(g) \cap \overline{z_1 z_2}$ be $v_1, v_2, \ldots, v_m$ ordered along $\overline{z_1 z_2}$ from z_1 to z_2, and let $v_0 = z_1$ and $v_{m+1} = z_2$. Under the condition of this lemma, since $t \in \mathcal{M}$, there exists a shortest path π from z to t that crosses $\overline{z_1 z_2}$ once, say, at a point p in the interior of $\overline{v_i v_{i+1}}$, for some i with $0 \leq i \leq m$ (see Fig. 12). For any two points q_1 and q_2 on π, let $\pi(q_1, q_2)$ denote the subpath of π between q_1 and q_2. Hence, $\pi(z, p)$ is in B and $\pi(p, t)$ is outside B. Then $\pi(p, t)$ is in $\mathcal{M}'$ (i.e., $\mathcal{M}'$ is the union of $\mathcal{M}$ and all corridor paths).

We extend a horizontal line segment from v_i (resp., v_{i+1}) to the right until hitting the first point on $\partial\mathcal{Q}$, denoted by u_i (resp., u_{i+1}); if u_i and u_{i+1} are not on the same elementary curve of $\mathcal{Q}$ (in which case one or both of u_i and u_{i+1} are extremes on different elementary curves), then we keep moving one or both of u_i and u_{i+1} horizontally to the right until hitting the next point on $\partial\mathcal{Q}$. By the definitions of $\mathcal{V}(\mathcal{Q})$ and $\mathcal{V}(g)$, in this way, we can always put both u_i and u_{i+1} on the same elementary curve of $\mathcal{Q}$, say β (see Fig. 12); let $\beta(u_i, u_{i+1})$ denote the portion of β between u_i and u_{i+1}. Let R denote the region enclosed by $\overline{u_i v_i}$, $\overline{v_i v_{i+1}}$, $\overline{v_{i+1} u_{i+1}}$, and $\beta(u_i, u_{i+1})$. Note that for any point $q \in R$, q is horizontally visible to $\overline{v_i v_{i+1}}$ and thus is horizontally visible to $\overline{z z_2}$. In the following, we will show that t must be in R, which proves the lemma.

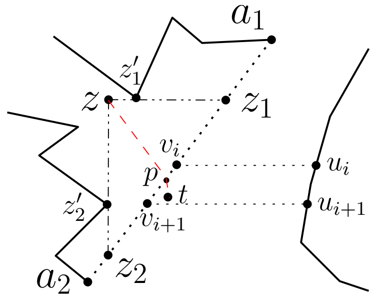

Figure 12: Illustrating a shortest path (the red dashed curve) from z to t crossing the interior of $\overline{v_iv_{i+1}}$ at p.

Suppose to the contrary $t \notin R$. We then show that the path $\pi(p,t)$ must intersect $\overline{v_iu_i}$ or $\overline{v_{i+1}u_{i+1}}$, which implies that there is a shortest z-t path containing a point in $\{z_1, z_2\} \cup \{\mathcal{V}(g) \cap \overline{z_1z_2}\}$, a contradiction (recall that we have an assumption that no shortest s-t paths cross $\overline{a_1z_1} \cup \overline{a_2z_2}$). Indeed, if $\pi(p,t)$ intersects $\overline{v_iu_i}$ (resp., $\overline{v_{i+1}u_{i+1}}$), say, at a point q, then we can obtain a new z-t path π' by replacing $\pi(z,q)$ with an xy-monotone path $\overline{zv_i} \cup \overline{v_iq}$ (resp., $\overline{zv_{i+1}} \cup \overline{v_{i+1}q}$), and π' is a shortest z-t path containing a point in $\{z_1, z_2\} \cup \{\mathcal{V}(g) \cap \overline{z_1z_2}\}$. Below, we show that $\pi(p,t)$ must intersect $\overline{v_iu_i}$ or $\overline{v_{i+1}u_{i+1}}$. Note that $\beta(u_i, u_{i+1})$ may overlap with a gate of a canal. Depending on whether $\beta(u_i, u_{i+1})$ overlaps with any canal gate, there are two possible cases.

1. If $\beta(u_i, u_{i+1})$ does not overlap with any canal gate, then since $t \in \mathcal{M}$, $t \notin R$, $p \in R$, and $\pi(p,t) \subseteq \mathcal{M}'$, if we go from t to p, we must enter R. The only place on the boundary of R we can cross to enter R is either $\overline{v_iu_i}$ or $\overline{v_{i+1}u_{i+1}}$. Hence, $\pi(t,p)$ must intersect $\overline{v_iu_i}$ or $\overline{v_{i+1}u_{i+1}}$.

2. If $\beta(u_i, u_{i+1})$ overlaps with a canal gate, say g_1, then one may wonder that $\pi(t,p)$ could enter the interior of R through g_1 without crossing any of $\overline{v_iu_i}$ and $\overline{v_{i+1}u_{i+1}}$. Since g_1 is a canal gate, one of g_1's endpoints, say, x, must be a corridor path terminal, and x may or may not be on $\beta(u_i, u_{i+1})$. If x is on $\beta(u_i, u_{i+1})$, then since x is in $\mathcal{V}(\mathcal{Q})$, x cannot be in the interior of $\beta(u_i, u_{i+1})$ and can only be at an endpoint of $\beta(u_i, u_{i+1})$. Let C be the canal that has g_1 as a gate, and $\pi(C)$ be the corridor path of C. If $\pi(t,p)$ enters the interior of R through g_1, then it must travel through the canal C, implying that $\pi(t,p) \subseteq \mathcal{M}'$ contains the corridor path $\pi(C)$. Since x is on $\pi(C)$, $\pi(t,p)$ contains x. If x is on $\beta(u_i, u_{i+1})$ (and thus is an endpoint of $\beta(u_i, u_{i+1})$), then x is one of u_i or u_{i+1}; hence, $\pi(t,p)$ intersects $\overline{v_iu_i}$ or $\overline{v_{i+1}u_{i+1}}$. Suppose now x is not on $\beta(u_i, u_{i+1})$. Then an endpoint of $\beta(u_i, u_{i+1})$, say, u_i, lies on g_1 (but $u_i \neq x$). Further, $\pi(t,p)$ goes through x, and then enters R, but without intersecting any of $\overline{v_iu_i}$ and $\overline{v_{i+1}u_{i+1}}$. Thus, $\pi(t,p)$ must cross some point q of g_1 to enter R. We can then replace the portion $\pi(x,q)$ of $\pi(t,p)$ by the segment $\overline{xq} \subseteq g_1$ to obtain a new shortest t-p path. Since u_i divides g_1 into two parts, one outside R and containing x and the other intersecting R and containing q, the segment $\overline{xq}$ contains u_i. Hence, the new shortest t-p path intersects $\overline{v_iu_i}$.

The lemma thus follows. □

By Lemma 16, if the condition of the lemma holds, then we can always find a trivial shortest path from z to t by shooting vertical and horizontal rays from z and t, respectively.

We have finished all possible cases for finding a shortest s-t path when $s \in B$ and $t \in \mathcal{M}$. The next lemma is concerned with computing the special points z_1, z_2, and z for any point s in B.

Lemma 17. *With a preprocessing of $O(n_B)$ time and space, the three special points z_1, z_2, and z can be found in $O(\log n)$ time for any query point s in B, where $n_B = |B|$.*

Proof. Consider any query point $s \in B$. To determine z_1, z_2, and z, based on our previous discussions, it suffices to compute the two points z_1' and z_2'. We only show how to design a data structure for computing z_1' since the solution for finding z_2' is similar. Note that $n_B \leq n$.

In the preprocessing, for each vertex v of B, we find whether v is horizontally visible to g, and if yes, mark v as an *h-vertex*. All h-vertices of B can be marked by computing the horizontal visibility of B from g in $O(n_B)$ time [29, 33]. Also, in $O(n_B)$ time, we compute the Euclidean shortest path tree T_1 from a_1 to all vertices of B and the corresponding shortest path map M_1 in B [21]; similarly, we compute the shortest path tree T_2 from a_2 and the corresponding shortest path map M_2.

For each vertex $v \in T_1$, we associate v with two special vertices: $\alpha_1(v)$ and $\alpha_2(v)$, defined as follows. The vertex $\alpha_1(v)$ is the first h-vertex on the path in T_1 from v to a_1 and $\alpha_2(v)$ is the child vertex of $\alpha_1(v)$ on the path in T_1 from v to a_1; if $\alpha_1(v) = v$, then $\alpha_2(v)$ does not exist and we set $\alpha_2(v) = nil$. Note that $\alpha_2(v)$ is not an h-vertex if it exists. The α vertices for all vertices in T_1 can be computed in $O(n_B)$ time by a depth-first search on T_1 starting at a_1. For each vertex $v \in T_2$, we compute only one special vertex for v, $\beta_1(v)$, which is the first h-vertex on the path in T_2 from v to a_2. The β vertices for all vertices of T_2 can also be computed in $O(n_B)$ time.

This finishes our preprocessing, which takes $O(n_B)$ time in total.

Below we find the point z_1' in $O(\log n)$ time. Let $\pi(s, a_1)$ and $\pi(s, a_2)$ be the Euclidean shortest paths in B from s to a_1 and a_2, respectively. For any point p, let $y(p)$ denote its y-coordinate.

By using the shortest path map M_1, we find the vertex, denoted by v, which directly connects to s on $\pi(s, a_1)$. Likewise, we find the vertex u that directly connects to s on $\pi(s, a_2)$ using M_2. Both v and u are found in $O(\log n)$ time. Depending on whether $v = u$, there are two main cases.

1. If $v = u$, then clearly $s \neq z'$. Let $v_1 = \alpha_1(v)$ and $u_1 = \beta_1(u)$. Note that v_1 and u_1 are available once we find v and u. Depending on whether $v_1 = u_1$, we further have two subcases.

   (a) If $v_1 = u_1$, then we claim $z' = v_1 = u_1$. Indeed, since z' is the last common vertex of $\pi(s, a_1)$ and $\pi(s, a_2)$ if we move on them from s, no vertex on $\pi(s, a_1) \cap \pi(s, a_2)$ can be horizontally visible to g except possibly z'. Because $v_1 = u_1$, $v_1 = u_1$ must be on $\pi(s,a_1)\cap\pi(s,a_2)$. Since $v_1=u_1$ is horizontally visible to g, $v_1=u_1=z'$ must hold.

By the definition of z'_1, the above claim implies $z'_1=z'=u_1=v_1$.

(b) If $v_1\neq u_1$, then an easy observation is $y(v_1)\geq y(u_1)$. Let $v_2=\alpha_2(v)$. Note that due to $u=v$ and $v_1\neq u_1$, $\alpha_2(v)$ exists.

If $y(v_2)>y(v_1)$, then the horizontal visibility of v_2 to g is "blocked" by the path $\pi(v_1,a_1)$ (e.g., see Fig. 11(a)). Thus we obtain $z'_1=v_1$.

If $y(v_2)\leq y(v_1)$, then the horizontal visibility of v_2 to g is "blocked" by the path $\pi(u_1,a_2)$ (e.g., see Fig. 11(b)). Thus we obtain that z'_1 is the horizontal projection of u_1 on the line segment $\overline{v_1v_2}$, which can be computed in $O(1)$ time.

2. If $v\neq u$, then $s=z'$. If s is horizontally visible to g (which can be determined in $O(\log n)$ time using the horizontal visibility decomposition of B), then $z'_1=s=z'$. Otherwise, let $v_1=\alpha_1(v)$ and $u_1=\beta_1(u)$. Depending on whether $v=v_1$, we further have two subcases.

(a) If $v\neq v_1$, then $\alpha_2(v)$ exists and we let $v_2=\alpha_2(v)$. Note that $\pi(s,a_1)$ is a convex chain.

Similar to the above discussion, if $y(v_2)>y(v_1)$, then we have $z'_1=v_1$; otherwise, z'_1 is the horizontal projection of u_1 on $\overline{v_1v_2}$.

(b) If $v=v_1$, then s connects directly to v_1 on $\pi(s,a_1)$. Similar to the above discussion, if $y(s)>y(v_1)$, then we have $z'_1=v_1$; otherwise, z'_1 is the horizontal projection of u_1 on $\overline{v_1s}$.

Therefore, we can find the point z'_1 in $O(\log n)$ time. The lemma thus follows. □

We have discussed all possible cases of finding a shortest s-t path when s is in a bay B and t is in the ocean $\mathcal{M}$, and in each case, we can obtain a shortest path in $O(\log n)$ time.

4.3.3 The Point t is in Another Bay

Let B_s be the bay containing s with gate g_s, and B_t be the bay containing t with gate g_t. In this case, any shortest s-t path must cross both g_s and g_t. The algorithm for this case is similar to the one for the case of $t\in\mathcal{M}$. Again, we need to consider different cases of how a shortest s-t path may cross different portions of both the gates g_s and g_t.

We define the points z_1, z_2, and z in B_s for s in the same way as before, but denote them by $z_1(s)$, $z_2(s)$, and $z(s)$ instead. Similarly, we define the corresponding three points $z_1(t)$, $z_2(t)$, and $z(t)$ in B_t for t. Based on our previous discussions, we have the following cases.

1. There is a shortest s-t path containing a point z_s in $\{z_1(s),z_2(s)\}$ and a point z_t in $\{z_1(t),z_2(t)\}$. Note that both z_s and z_t are on their bay gates and thus are in $\mathcal{M}$.

In this case, there must be a shortest s-t path that is a concatenation of a shortest path $\pi(s, z_s)$ from s to z_s in B_s, a shortest path $\pi(z_s, z_t)$ from z_s to z_t in $\mathcal{M}'$, and a shortest path $\pi(z_t, t)$ from z_t to t in B_t. The path $\pi(s, z_s)$ can be found by using $\mathcal{D}(B_s)$, i.e., the Euclidean two-point shortest path query data structure on B_s [20], and similarly, $\pi(z_t, t)$ can be found by using $\mathcal{D}(B_t)$. The path $\pi(z_s, z_t)$ can be found by using our data structure for $\mathcal{M}'$ in Lemma 8.

2. There is a shortest s-t path that contains $z(s)$ and a point z_t in $\{z_1(t), z_2(t)\}$.

   In this case, there must be a shortest s-t path that is a concatenation of a shortest s-$z(s)$ path $\pi(s, z(s))$ in B_s, a shortest $z(s)$-z_t path $\pi(z(s), z_t)$, and a shortest z_t-t path $\pi(z_t, t)$ in B_t. The path $\pi(s, z(s))$ (resp., $\pi(z_t, t)$) can be found by using $\mathcal{D}(B_s)$ (resp., $\mathcal{D}(B_t)$), and the path $\pi(z(s), z_t)$ can be found by using similar algorithms as discussed above since z_t is in $\mathcal{M}$.

3. There is a shortest s-t path that contains $z(t)$ and a point z_s in $\{z_1(s), z_2(s)\}$.

   This case is solved by using the similar approach as for Case 2 above.

4. There is a shortest s-t path that contains $z(s)$ and $z(t)$.

   In this case, there must be a shortest s-t path that is a concatenation of a shortest path $\pi(s, z(s))$ from s to $z(s)$ in B_s, a shortest path $\pi(z(s), z(t))$ from $z(s)$ to $z(t)$, and a shortest path $\pi(z(t), t)$ from $z(t)$ to t in B_t. The path $\pi(s, z(s))$ (resp., $\pi(z(t), t)$) can be found by using $\mathcal{D}(B_s)$ (resp., $\mathcal{D}(B_t)$). It remains to show how to compute $\pi(z(s), z(t))$ below.

Recall that we have defined a graph $G_E(g_s)$ in B_s on the points of $\mathcal{V}(g_s)$, which consists of all points of $\mathcal{V}(\mathcal{Q})$ lying on g_s. We also find a gateway set $V_g(z(s), G_E(g_s))$ for $z(s)$ on $G_E(g_s)$. Similarly, for B_t and its gate g_t, we define $\mathcal{V}(g_t)$, $G_E(g_t)$, and $V_g(z(t), G_E(g_t))$. Let $G_E(\mathcal{M}, g_s, g_t)$ be the graph formed by merging $G_E(\mathcal{M})$, $G_E(g_s)$, and $G_E(g_t)$. A shortest path from $z(s)$ to $z(t)$ can be found based on Lemmas 18 and 19 below, which are similar to Lemmas 14 and 16, respectively.

Lemma 18. *If there is a shortest path from $z(s)$ to $z(t)$ containing a point in $\mathcal{V}(g_s) \cap \overline{z_1(s)z_2(s)}$ and a point in $\mathcal{V}(g_t) \cap \overline{z_1(t)z_2(t)}$, then there is a shortest path from $z(s)$ to $z(t)$ along $G_E(\mathcal{M}, g_s, g_t)$ that contains a gateway of $z(s)$ in $V_g(z(s), G_E(g_s))$ and a gateway of $z(t)$ in $V_g(z(t), G_E(g_t))$.*

Proof. Suppose there is a shortest $z(s)$-$z(t)$ path containing a point p_s in $\mathcal{V}(g_s) \cap \overline{z_1(s)z_2(s)}$ and a point p_t in $\mathcal{V}(g_t) \cap \overline{z_1(t)z_2(t)}$. Then by Lemma 13, there is a shortest $z(s)$-p_s path $\pi(z(s), p_s)$ along $G_E(g_s)$ containing a gateway of $z(s)$ in $V_g(z(s), G_E(g_s))$ and there is a shortest p_t-$z(t)$ path $\pi(p_t, z(t))$ along $G_E(g_t)$ containing a gateway of $z(t)$ in $V_g(z(t), G_E(g_t))$. Since both p_s and p_t are in $\mathcal{V}(\mathcal{Q})$, by Lemma 4, there exists a shortest p_s-p_t path $\pi(p_s, p_t)$ along $G_E(\mathcal{M})$.

The concatenation of $\pi(z(s), p_s)$, $\pi(p_s, p_t)$, and $\pi(p_t, z(t))$ is a shortest $z(s)$-$z(t)$ path, which is along the graph $G_E(\mathcal{M}, g_s, g_t)$ and contains a gateway of $z(s)$ and a gateway of $z(t)$. □

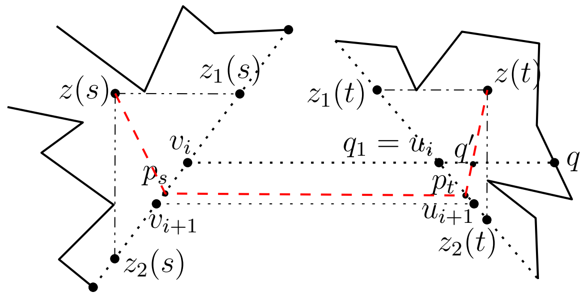

Figure 13: Illustrating a possible shortest path (the red dashed curve) from $z(s)$ to $z(t)$ crossing $\overline{u_iq}$. If this happens, we can find another shortest path $\pi(z(t), q') \cup \overline{q'v_i} \cup \overline{v_iz(s)}$, which contains v_i. In this example, $q_1 = u_i$ and $q_2 = u_{i+1}$.

Lemma 19. *If no shortest $z(s)$-$z(t)$ path contains any point of $\{z_1(s), z_2(s)\} \cup \{\mathcal{V}(g_s) \cap \overline{z_1(s)z_2(s)}\}$, then $z(t)$ must be horizontally visible to $\overline{z(s)z_2(s)}$; similarly, if no shortest $z(s)$-$z(t)$ path contains any point of $\{z_1(t), z_2(t)\} \cup \{\mathcal{V}(g_t) \cap \overline{z_1(t)z_2(t)}\}$, then $z(s)$ must be horizontally visible to $\overline{z(t)z_2(t)}$.*

Proof. We prove only the case where no shortest $z(s)$-$z(t)$ path contains any point of $\{z_1(s), z_2(s)\} \cup \{\mathcal{V}(g_s) \cap \overline{z_1(s)z_2(s)}\}$, $z(t)$ must be horizontally visible to $\overline{z(s)z_2(s)}$ (the other case is similar).

Let π be a shortest $z(s)$-$z(t)$ path that intersects g_s at a point p_s and intersects g_t at a point p_t (see Fig. 13). Let $\pi(p_1, p_2)$ denote the subpath of π between any two points p_1 and p_2 on π. We assume $\pi(z(s), p_s) \subseteq B_s$, $\pi(p_t, z(t)) \subseteq B_t$, and $\pi(p_s, p_t) \subseteq \mathcal{M}'$, since such a path π always exists.

Let the points of $\mathcal{V}(g_s)$ on $\overline{z_1(s)z_2(s)}$ be $v_1, v_2, \ldots, v_m$ ordered along $\overline{z_1(s)z_2(s)}$ from $z_1(s)$ to $z_2(s)$, and let $v_0 = z_1(s)$ and $v_{m+1} = z_2(s)$. Suppose p_s is in the interior of $\overline{v_iv_{i+1}}$, for some i with $0 \le i \le m$. We define u_i, u_{i+1}, $\beta(u_i, u_{i+1})$, and R in the same way as in the proof of Lemma 16.

Since p_t is in $\mathcal{M}$, by the proof of Lemma 16, p_t must be in the region R. Further, since p_t is on $g_t \subseteq \partial\mathcal{Q}$, p_t is on $\beta(u_i, u_{i+1})$. Thus, $\beta(u_i, u_{i+1}) \cap g_t$ is not empty. Since g_t is a line segment, $\beta(u_i, u_{i+1}) \cap g_t$ is also a line segment. Let $\overline{q_1q_2} = \beta(u_i, u_{i+1}) \cap g_t$. Thus, $p_t \in \overline{q_1q_2}$.

Recall that $z(t)$ is visible to $z_1(t) \in g_t$ and $\overline{z(t)z_1(t)}$ is horizontal. Hence, to prove that $z(t)$ is horizontally visible to $\overline{z(s)z_2(s)}$, it suffices to prove that $z_1(t)$ is horizontally visible to $\overline{z(s)z_2(s)}$. For this, it suffices to prove that $z_1(t)$ must be on $\overline{q_1q_2}$ since every point on $\overline{q_1q_2} \subseteq \beta(u_i, u_{i+1})$ is horizontally visible to $\overline{z(s)z_2(s)}$. In the following, we prove $z_1(t) \in \overline{q_1q_2}$.

Suppose to the contrary $z_1(t) \notin \overline{q_1q_2}$ (see Fig. 13). Without loss of generality, we assume q_1 is closer to $z_1(t)$ than q_2. Since $p_t \in \overline{q_1q_2}$, $q_1 \in \overline{p_tz_1(t)} \subseteq g_t$. This implies that q_1 is not an endpoint of g_t, and thus q_1 must be an endpoint of $\beta(u_i, u_{i+1})$ (i.e., one of u_i or u_{i+1}) since $\overline{q_1q_2} = \beta(u_i, u_{i+1}) \cap g_t$; assume $q_1 = u_i$. We extend $\overline{v_iu_i}$ horizontally into the bay B_t until hitting a point, say q, on the boundary of B_t (see Fig. 13). The horizontal segment $\overline{u_iq}$ partitions B_t into two sub-polygons such that p_t and $z_1(t)$ are in different sub-polygons.

Since $\overline{z(t)z_1(t)}$ is horizontal, p_t and $z(t)$ are also in different sides of $\overline{u_iq}$, implying that the path $\pi(z(t), p_t)$ must intersect $\overline{u_iq}$ since $\pi(z(t), p_t)$ is in B_t. Let q' be the intersection of $\pi(z(t), p_t)$ and $\overline{u_iq}$ (see Fig. 13). Then, the concatenation of $\pi(z(t), q')$, $\overline{q'v_i}$, and $\overline{v_iz(s)}$ is also a shortest path from $z(t)$ to $z(s)$ since $\overline{q'v_i} \cup \overline{v_iz(s)}$ is xy-monotone. But this means that there is a shortest $z(s)$-$z(t)$ path containing v_i, contradicting with the lemma condition that no shortest $z(s)$-$z(t)$ path contains any point of $\{z_1(s), z_2(s)\} \cup \{\mathcal{V}(g_s) \cap \overline{z_1(s)z_2(s)}\}$.

The above arguments prove that $z_1(t)$ is on $\overline{q_1q_2}$. The lemma thus follows. □

By Lemmas 18 and 19, we can find a shortest $z(s)$-$z(t)$ path by either using the gateways of $z(s)$ and $z(t)$ in the merged graph $G_E(\mathcal{M}, g_s, g_t)$ or shooting horizontal and vertical rays from $z(s)$ and $z(t)$. We have finished all possible cases for finding a shortest s-t path when the two query points are in different bays. For each case, we compute a "candidate" shortest s-t path, and take the one with the smallest length among all these cases (there are only a constant number of them).

4.3.4 The Canal Case

The above discussed the case where neither query point is in a canal. It remains to solve the canal case where at least one of the query points are in canals.

The algorithm is similar to that for the bay case; the only difference is that we have to take care of two gates for each canal. Specifically, we consider the most general case where s is in a canal C_s and t is in a canal C_t. If $C_s \neq C_t$, then there must be a shortest s-t path π that intersects a gate of C_s at a point p_s and intersects a gate of C_t at a point p_t such that the subpath $\pi(s, p_s)$ is in C_s, the subpath $\pi(p_t, t)$ is in C_t, and the subpath $\pi(p_s, p_t)$ is in $\mathcal{M}'$. Hence, we can use a similar approach as for the bay case to find a shortest s-t path by considering all four gate pairs of C_s and C_t. If $C_s = C_t$, while we can treat this case in the same way as for the case of $C_s \neq C_t$, we need to consider one more possible situation when a shortest s-t path may be contained entirely in C_s, which is easy since C_s is a simple polygon. If one of C_s or C_t is a bay, the case can be handled in a similar fashion.

We summarize the whole algorithm in the proof of the following theorem.

Theorem 2. *We can build a data structure of size $O(n + h^2 \log h 4^{\sqrt{\log h}})$ in time $O(n + h^2 \log^2 h 4^{\sqrt{\log h}})$ such that each two-point L_1 shortest path query can be answered in $O(\log n)$ time (i.e., for any two query points s and t, the length of a shortest s-t path can be found in $O(\log n)$ time and an actual path can be reported in additional time linear in the number of edges of the output path).*

Proof. We first discuss the preprocessing and then discuss the query algorithm.

The preprocessing algorithm. Our preprocessing algorithm consists of the following major steps.

1. Compute a triangulation of the free space $\mathcal{M}$ in $O(n + h \log^{1+\epsilon} h)$ time [2, 4]. Then produce all bays, canals, corridor paths, $\mathcal{M}$, and $\mathcal{V}(\mathcal{Q})$ in $O(n+h \log h)$ time [8, 9, 11].

2. Compute the vertical and horizontal visibility decompositions of $\mathcal{P}$ in $O(n+h\log^{1+\epsilon} h)$ time [2, 4]. Build a point location data structure [18, 32] for each of the two decompositions in $O(n)$ time, which is used for performing any vertical or horizontal ray-shooting in $O(\log n)$ time.

3. Construct the graph $G_E(\mathcal{M})$ of size $O(n+h\sqrt{\log h}2^{\sqrt{\log h}})$ in $O(n+h\log^{3/2} h2^{\sqrt{\log h}})$ time by Lemma 5.

4. Perform the preprocessing of Lemma 7 in $O(n+h\cdot\log^{3/2} h\cdot 2^{\sqrt{\log h}})$ time and $O(n+h\cdot\sqrt{\log h}\cdot 2^{\sqrt{\log h}})$ space.

5. Perform the preprocessing of Lemma 8 in $O(n+h^2\log^2 h4^{\sqrt{\log h}})$ time and $O(n+h^2\log h4^{\sqrt{\log h}})$ space.

6. Compute a two-point Euclidean shortest path query data structure $\mathcal{D}(B)$ in each bay or canal B. Since the total number of vertices of all bays and canals is $O(n)$, this step takes $O(n)$ time.

7. Construct the graph $G_E(g)$ for the gate g of every bay or canal by Lemma 12. The total space for all such graphs is $O(h\sqrt{\log h}2^{\sqrt{\log h}})$ and the total time for building all these graphs is $O(n+h\log^{3/2} h2^{\sqrt{\log h}})$, as proved below. First, each point of $\mathcal{V}(\mathcal{Q})$ can be on at most one bay or canal gate. Thus, the sum of h_g's in Lemma 12 over all gates g is $O(|\mathcal{V}(\mathcal{Q})|)$, which is $O(h)$. Second, the total number of obstacle vertices of all bays and canals is $O(n)$, and each canal has two gates. Hence, the sum of n_B's in Lemma 12 over all bays and canals B is $O(n)$.

8. Perform the preprocessing of Lemma 15 for the graphs $G_E(g)$ of all gates g, which can be done in totally $O(h\log^{3/2} h2^{\sqrt{\log h}})$ time and $O(h\sqrt{\log h}2^{\sqrt{\log h}})$ space.

9. Merge the graph $G_E(\mathcal{M})$ and the graphs $G_E(g)$ for all gates g into a single graph $G_E(\mathcal{P})$, which takes $O(h)$ time since there are $O(h)$ points in $\mathcal{V}(\mathcal{Q})$. Thus, the size of $G_E(\mathcal{P})$ is $O(h\sqrt{\log h}2^{\sqrt{\log h}})$.

10. For each node v of $G_E(\mathcal{P})$, compute a shortest path tree rooted at v in $G_E(\mathcal{P})$. Maintain a shortest path length table such that for any two nodes u and v of $G_E(\mathcal{P})$, the length of a shortest path between u and v in $G_E(\mathcal{P})$ can be obtained in $O(1)$ time. This step takes $O(h^2\log h4^{\sqrt{\log h}})$ space and $O(h^2\log^2 h4^{\sqrt{\log h}})$ time.

11. Perform the preprocessing of Lemma 17 for each bay and canal, which takes $O(n)$ space and $O(n)$ time in total.

In summary, the total preprocessing space and time are $O(n+h^2\log h4^{\sqrt{\log h}})$ and $O(n+h^2\log^2 h4^{\sqrt{\log h}})$, respectively.

The query algorithm. Consider any two query points s and t. Next, we discuss our query algorithm that computes the length of a shortest s-t path in $O(\log n)$ time and reports an actual path in additional time linear in the number of edges of the output path. We will not explicitly discuss how to report an actual path (which is similar to that in Lemma 8 and is quite straightforward).

First of all, as discussed in Section 2, we determine whether there exists a trivial shortest s-t path by shooting horizontal and vertical rays from s and t, which can be done in $O(\log n)$ time. In the following, we assume that there is no trivial shortest s-t path. Depending on whether the query points are in the bays, canals, or the ocean $\mathcal{M}$, there are several possible cases.

I. Both query points are in $\mathcal{M}$. In this case, we use the algorithm for Lemma 8 to find a shortest s-t path in $O(\log n)$ time.

II. Only one query point is in $\mathcal{M}$. Without loss of generality, we assume that s is in a bay or a canal B and t is in $\mathcal{M}$. Further, we assume that B is a canal since the case that B is a bay can be considered as a special case.

Let g^1 and g^2 be the two gates of B. We define three points $z(s, g^1)$, $z_1(s, g^1)$, and $z_2(s, g^1)$ for s in B with respect to the gate g^1 in the same way as we defined z, z_1, and z_2 before. Similarly, we define $z(s, g^2)$, $z_1(s, g^2)$, and $z_2(s, g^2)$ for s in B with respect to the gate g^2. These points can be computed in $O(\log n)$ time by Lemma 17. Then, we compute the lengths of the following "candidate" shortest s-t paths and return the one with the smallest length.

1. For each point $p \in \{z_1(s, g^1), z_2(s, g^1), z_1(s, g^2), z_2(s, g^2)\}$, the path which is a concatenation of a shortest path $\pi(s, p)$ from s to p in B and a shortest path $\pi(p, t)$ from p to t in $\mathcal{M}'$.

   The path $\pi(s, p)$ can be found in $O(\log n)$ time by using the data structure $\mathcal{D}(B)$ on B, and the path $\pi(p, t)$ can be found in $O(\log n)$ time by Lemma 8.

2. For each point $p \in \{z(s, g^1), z(s, g^2)\}$, the path which is a concatenation of a shortest path $\pi(s, p)$ from s to p in B and a particular path $\pi(p, t)$ from p to t.

   The path $\pi(s, p)$ can be found in $O(\log n)$ time by using the data structure $\mathcal{D}(B)$ on B. The path $\pi(p, t)$ is determined as follows. First, based on Lemma 16 (although B is a bay in Lemma 16, the result also holds for canals because the lemma was proved with respect to a gate regardless of whether it is a gate of a bay or a canal), we check whether there exists a path from p to t consisting of only two line segments, by performing horizontal and vertical ray-shootings. If yes, then such a path is $\pi(p, t)$. Otherwise, by Lemmas 14 and 16, we find a shortest path from p to t along the merged graph $G_E(\mathcal{P})$ by using the gateways of p and the gateways of t, which can be obtained in $O(\log n)$ time by Lemmas 15 and 7, respectively. Since both p and t have $O(\sqrt{\log h})$ gateways, a shortest p-t path can be determined in $O(\log n)$ time using the gateway graph as discussed at the end of Section 3.

III. Neither query point is in $\mathcal{M}$. Let B_s be the bay or canal that contains s and B_t be the bay or canal that contains t.

1. If $B_s = B_t$ and B_s is a bay, then by Lemma 9, we can find a shortest s-t path by using the data structure $\mathcal{D}(B_s)$ in $O(\log n)$ time.
2. If $B_s \neq B_t$, then we assume both B_s and B_t are canals since the other cases are just special cases of this case. Let g_s^1 and g_s^2 be the two gates of B_s and g_t^1 and g_t^2 be the two gates of B_t. Similarly as before, we define the points $z(s, g_s^i)$, $z_1(s, g_s^i)$, and $z_2(s, g_s^i)$ for s with respect to g_s^i, and $z(t, g_t^i)$, $z_1(t, g_t^i)$, and $z_2(t, g_t^i)$ for t with respect to g_t^i, for $i = 1, 2$. These points can all be determined in $O(\log n)$ time by Lemma 17. Then we compute the lengths of the following "candidate" shortest s-t paths and return the one with the smallest length.
   (a) For each pair of points p_s and p_t such that $p_s \in \{z_1(s, g_s^1), z_2(s, g_s^1), z_1(s, g_s^2), z_2(s, g_s^2)\}$ and $p_t \in \{z_1(t, g_t^1), z_2(t, g_t^1), z_1(t, g_t^2), z_2(t, g_t^2)\}$, the path which is a concatenation of a shortest path $\pi(s, p_s)$ from s to p_s in B_s, a shortest path from p_s to p_t in $\mathcal{M}'$, and a shortest path $\pi(p_t, t)$ from p_t to t in B_t.
   The paths $\pi(s, p_s)$ and $\pi(p_t, t)$ can be found in $O(\log n)$ time by using $\mathcal{D}(B_s)$ and $\mathcal{D}(B_t)$, respectively. The path $\pi(p_s, p_t)$ can be obtained in $O(\log n)$ time by Lemma 8.
   (b) For each pair of points p_s and p_t such that $p_s \in \{z(s, g_s^1), z(s, g_s^2)\}$ and $p_t \in \{z_1(t, g_t^1), z_2(t, g_t^1), z_1(t, g_t^2), z_2(t, g_t^2)\}$, the path which is a concatenation of a shortest path from s to p_s in B_s, a particular path $\pi(p_s, p_t)$ from p_s to p_t, and a shortest path from p_t to t in B_t.
   The paths $\pi(s, p_s)$ and $\pi(p_t, t)$ can be found in $O(\log n)$ time by using $\mathcal{D}(B_s)$ and $\mathcal{D}(B_t)$, respectively. Since p_t is in $\mathcal{M}$, the particular path $\pi(p_s, p_t)$ is defined similarly as the path $\pi(p, t)$ in the above Case (II)2, and thus can be obtained by the similar approach.
   (c) For each point $p_s \in \{z_1(s, g_s^1), z_2(s, g_s^1), z_1(s, g_s^2), z_2(s, g_s^2)\}$ and each point $p_t \in \{z(t, g_t^1), z(t, g_t^2)\}$, the path which is a concatenation of a shortest path from s to p_s in B_s, a particular path $\pi(p_s, p_t)$ from p_s to p_t, and a shortest path from p_t to t in B_t.
   This subcase is symmetric to the subcase immediately above and can be handled similarly.
   (d) For each point $p_s \in \{z(s, g_s^1), z(s, g_s^2)\}$ and each point $p_t \in \{z(t, g_t^1), z(t, g_t^2)\}$, the path which is a concatenation of a shortest path from s to p_s in B_s, a particular path $\pi(p_s, p_t)$ from p_s to p_t, and a shortest path from p_t to t in B_t.
   The paths $\pi(s, p_s)$ and $\pi(p_t, t)$ can be found in $O(\log n)$ time by using $\mathcal{D}(B_s)$ and $\mathcal{D}(B_t)$, respectively. The particular path $\pi(p_s, p_t)$ is determined similarly as the path $\pi(p, t)$ in the above Case (II)2, but based on Lemmas 18 and 19 instead. Note that although B_s and B_t are bays in these lemmas, the results also hold for canals (actually, they are proved with respect to two gates regardless of whether they are gates of bays or canals). Specifically, we determine $\pi(p_s, p_t)$ as follows. Based on Lemma 19, we first check whether there exists a path from p_s to p_t consisting of only two line segments, by horizontal and vertical ray-shootings. If yes, then such a path is $\pi(p_s, p_t)$. Otherwise, by Lemmas 18 and 19, we find a shortest p_s-p_t path along the

merged graph $G_E(\mathcal{P})$ by using the gateways of p_s and the gateways of p_t, which can be computed in $O(\log n)$ time by Lemma 15. Since both p_s and p_t have $O(\sqrt{\log h})$ gateways, a shortest p_s-p_t path can be obtained in $O(\log n)$ time using the gateway graph as discussed in Section 3.

3. Finally, if $B_s = B_t$ and B_s is a canal, then the algorithm is similar as for the above Case (III)2, with the difference that we must consider an additional "candidate" path that is a shortest s-t path inside B_s, which can be found in $O(\log n)$ time by using the data structure $\mathcal{D}(B_s)$.

Hence, in any case, we find a shortest s-t path in $O(\log n)$ time.

The theorem thus follows. □

If we replace all enhanced graphs, e.g., $G_E(\mathcal{M})$ and $G_E(g)$ for every gate g, by the corresponding graphs similar to G_{old} in [7] as discussed in Section 2, then we obtain the following results.

Corollary 3. *We can build a data structure in $O(n + h^2 \log^2 h)$ time and space, such that each two-point L_1 shortest path query is answered in $O(\log n + \log^2 h)$ time; alternatively, we can build a data structure in $O(nh \log h + h^2 \log^2 h)$ time and $O(nh \log h)$ space, such that each two-point L_1 shortest path query is answered in $O(\log n \log h)$ time.*

Proof. If we replace all the enhanced graphs $G_E(\mathcal{M})$ and $G_E(g)$ for every gate g of the bays and canals by the graphs similar to G_{old} in [7] as discussed in Section 2, then the size of the new merged graph, denoted by $G_{old}(\mathcal{P})$, becomes $O(h \log h)$ instead of $O(h\sqrt{\log h}2^{\sqrt{\log h}})$. Hence, the data structure for Theorem 2 needs $O(n + h^2 \log^2 h)$ space and can be built in $O(n + h^2 \log^2 h)$ time by using the approach in [7]. However, using the new graph $G_{old}(\mathcal{P})$, each query for any two points in $\mathcal{M}$ can be answered in $O(\log^2 h)$ time because there are $O(\log h)$ gateways for each query point. Therefore, any general two-point shortest path query can be answered in $O(\log^2 h + \log n)$ time, by using a similar query algorithm as in Theorem 2. We omit the details.

In the result above, we compute a shortest path tree rooted at each node in the merged graph $G_{old}(\mathcal{P})$. Alternatively, we can compute a shortest path map in the free space $\mathcal{F}$ for each node v of $G_{old}(\mathcal{P})$, such that given any query point t, the length of a shortest path from v to t can be found in $O(\log n)$ time and an actual path can be reported in additional time linear in the number of edges of the output path. Each such shortest path map is of size $O(n)$ and can be computed in $O(n + h \log h)$ time [8, 9, 11] (after the free space $\mathcal{F}$ is triangulated). Since the size of $G_{old}(\mathcal{P})$ is $O(h \log h)$, the overall preprocessing time and space are $O(nh \log h + h^2 \log^2 h)$ and $O(nh \log h)$, respectively. For answering queries, since a query point may have $O(\log h)$ gateways and for each gateway v, we can determine the shortest path from v to the other query point in $O(\log n)$ time, the total query time is $O(\log h \log n)$. We omit the details. □

5 The Weighted Rectilinear Case

In this section, we extend our techniques in Section 3 to the weighted rectilinear case. In the weighted rectilinear case, every polygonal obstacle $P \in \mathcal{P}$ is *rectilinear* and *weighted*, i.e., each edge of P is either horizontal or vertical and P has a weight $w(P) \geq 0$ ($w(P) = +\infty$ is possible). If a line segment e is in P, then the *weighted length* of e is $x \cdot (1 + w(P))$, where x is the L_1 length of e. Any polygonal path π can be divided into a sequence of maximal line segments such that each segment is contained in the same obstacle or in the free space $\mathcal{F}$; the *weighted length* of π is the sum of the weighted lengths of all maximal line segments of π.

Consider a vertex v of any rectilinear obstacle P such that the interior angle of P at v is $3\pi/2$ (e.g., see Fig. 14). We define the *internal projections* of v on the boundary ∂P of P as follows. Suppose $\overline{u_1v}$ and $\overline{u_2v}$ are the two edges of P incident to v. We extend $\overline{u_1v}$ into the interior of P along the direction from u_1 to v until we hit ∂P at the first point, which is an *internal projection* of v; similarly, we define another internal projection of v by extending $\overline{u_2v}$. Internal projections are used to control shortest paths that pass through the interior of obstacles.

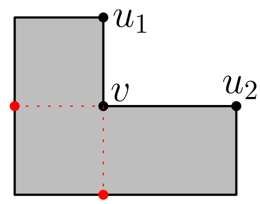

Figure 14: Illustrating the internal projections (the two red points) of v.

The "visibility" in the weighted case is defined in a slightly different way: Two points p and q are *visible* to each other if $\overline{pq}$ is entirely in either $\mathcal{F}$ or an obstacle.

Let $\mathcal{V}$ be the set of all obstacle vertices of $\mathcal{P}$, their internal projections, and all type-1 Steiner points. Then $|\mathcal{V}| = O(n)$. We build a graph $G_E(\mathcal{V})$ on $\mathcal{V}$ similar to the one presented in Section 3, with the following differences. (1) The visibility here is based on the new definition above. (2) Since a path can travel through the interior of any obstacle, for each cut-line l, an edge in $G_E(\mathcal{V})$ connects every two consecutive Steiner points on l, whose weight is the weighted length of the line segment connecting the two points. (3) In addition to the vertical cut-lines, there are also horizontal cut-lines, which are defined similarly and have type-2 and type-3 Steiner points defined on them similarly to those on the vertical cut-lines. Thus, $G_E(\mathcal{V})$ has $O(n\sqrt{\log n}2^{\sqrt{\log n}})$ nodes and edges.

Lemma 20. *The graph of $G_E(\mathcal{V})$ can be built in $O(n \log^{3/2} n 2^{\sqrt{\log n}})$ time.*

Proof. We obtain all internal projections of $\mathcal{V}$ by computing the horizontal and vertical visibility decompositions of every obstacle in $\mathcal{P}$. We find the four projection points on $\partial\mathcal{P}$ (i.e., p^r, p^l, p^u, and p^d) for all obstacle vertices p of $\mathcal{P}$ in $O(n \log n)$ time by computing the horizontal and vertical visibility decompositions of $\mathcal{F}$. All these can be done in $O(n \log n)$ time.

Then we compute the vertical and horizontal cut-line trees, which takes $O(n \log n)$ time since $|\mathcal{V}| = O(n)$. Next, we compute the Steiner points and the graph edges. Below, we only show how to compute those related to the vertical cut-lines; those related to the horizontal cut-lines can be computed in a similar way. Let $T^v(\mathcal{V})$ denote the vertical cut-line tree.

As in Lemma 1, we can compute the type-2 and type-3 Steiner points on all cut-lines of $T^v(\mathcal{V})$ by traversing $T^v(\mathcal{V})$ in a top-down manner. Since the internal projections and $\{p^r, p^l, p^u, p^d\}$ for each obstacle vertex p have been obtained, we can compute all $O(n\sqrt{\log n}2^{\sqrt{\log n}})$ such Steiner points in $O(n\sqrt{\log n}2^{\sqrt{\log n}})$ time; the corresponding horizontal graph edges connecting these Steiner points and the points of $\mathcal{V}$ can also be computed.

It remains to compute the graph edges connecting every pair of consecutive Steiner points on each cut-line of $T^v(\mathcal{V})$, which takes $O(n \log^{3/2} n 2^{\sqrt{\log n}})$ time by a plane sweeping algorithm, as follows. We first sort all Steiner points on each cut-line. We then sweep a vertical line L from left to right and use a balanced binary search tree T to maintain the intervals between the obstacle edges of $\mathcal{P}$ intersecting L. By standard techniques, we augment T to also maintain the weighted length information along L such that for any two points p and q on L, the weighted length of $\overline{pq}$ can be obtained in $O(\log n)$ time using T. During the sweeping, when L encounters a cut-line l, for every two consecutive Steiner points p and q on l, we use T to determine in $O(\log n)$ time the weighted length of the edge connecting p and q. Since there are $O(n\sqrt{\log n}2^{\sqrt{\log n}})$ pairs of consecutive Steiner points on all cut-lines, it takes $O(n \log^{3/2} n 2^{\sqrt{\log n}})$ time to compute all these graph edges.

Hence, we can build the graph $G_E(\mathcal{V})$ in $O(n \log^{3/2} n 2^{\sqrt{\log n}})$ time. □

Consider any two query points s and t. For simplicity of discussion, we assume that both s and t are in $\mathcal{F}$ (the general case can also be handled similarly). With a preprocessing of $O(n^2)$ time and space, a shortest s-t path that does not contain any vertex of $\mathcal{V}$ can be found in $O(\log n)$ time [7]. Thus in the following, we focus on finding a shortest s-t path containing at lease one vertex of $\mathcal{V}$.

Let $Y(s)$ be the set consisting of s and the four projections of s on $\partial\mathcal{P}$, i.e., $Y(s) = \{s, s^l, s^r, s^u, s^d\}$; similarly, let $Y(t) = \{t, t^l, t^r, t^u, t^d\}$. It was shown in [7] that it suffices to find a shortest path from p to q containing a vertex of $\mathcal{V}$ for every $p \in Y(s)$ and every $q \in Y(t)$. With a little abuse of notation, we let s be any point in $Y(s)$ and t be any point in $Y(t)$. Our goal is to find a shortest s-t path that contains at lease one vertex of $\mathcal{V}$. Unless otherwise indicated, any shortest s-t path mentioned below refers to a shortest s-t path that contains a vertex of $\mathcal{V}$.

In [7], similar to the discussions in Section 2, $O(\log n)$ gateways for s and $O(\log n)$ gateways for t were defined, such that any shortest s-t path must contain a gateway of s and a gateway of t. Hence by using the gateway graph, a shortest s-t path can be found in $O(\log^2 n)$ time.

Based on our enhanced graph $G_E(\mathcal{V})$, as in Section 3, we define a new gateway set $V_g(s, G_E(\mathcal{V}))$ of size $O(\sqrt{\log n})$ for s and a new gateway set $V_g(t, G_E(\mathcal{V}))$ of size $O(\sqrt{\log n})$ for t. The gateway set $V_g(s, G_E(\mathcal{V}))$ contains $O(\sqrt{\log n})$ Steiner points on the vertical cut-lines defined in the same way as those in $V_g^2(s, G_E)$ in Section 3; similarly, $V_g(s, G_E(\mathcal{V}))$ also contains $O(\sqrt{\log n})$ Steiner points on the horizontal cut-lines. The gateway set $V_g(t, G_E(\mathcal{V}))$ is defined similarly. Using a similar proof as for Lemma 2, we can show that there exists a shortest s-t path containing a gateway of s in $V_g(s, G_E(\mathcal{V}))$ and a gateway of t in $V_g(t, G_E(\mathcal{V}))$. Next, we show how to compute the two gateway sets and (the weights of) their gateway edges. Below, we discuss only the case for s.

The fractional cascading approach [5] used in Section 3 can still compute the gateway set $V_g(s, G_E(\mathcal{V}))$ in $O(\log n)$ time, but it cannot compute the weights of the gateway edges in $O(\log n)$ time for the following reasons. Consider a gateway $v \in V_g(s, G_E(\mathcal{V}))$, say on a vertical cut-line l. Then there is a gateway edge (s, v) that consists of two line segments $\overline{ss_h(l)}$ and $\overline{s_h(l)v}$ (recall that $s_h(l)$ is the horizontal projection of s on l). Hence, the weighted length of the edge (s, v) is the sum of the weighted lengths of these two line segments. It was shown in [7] that $\overline{ss_h(l)}$ must be in the free space (since s is in $\mathcal{F}$); thus, the weighted length of $\overline{ss_h(l)}$ is easy to compute. However, the vertical segment $\overline{s_h(l)v}$ may intersect multiple obstacles [7]. We give an algorithm to compute in $O(\log n)$ time the gateways and the weights of the gateway edges for s in the next lemma.

Lemma 21. *With a preprocessing of $O(n^2 \log n)$ time and $O(n^2)$ space, the gateways of $V_g(s, G_E(\mathcal{V}))$ for s and their weighted edges can be computed in $O(\log n)$ time.*

Proof. We discuss only how to compute the gateways of $V_g(s, G_E(\mathcal{V}))$ that are on the vertical cut-lines since those on the horizontal cut-lines can be computed similarly. Further, we only compute the gateways of $V_g(s, G_E(\mathcal{V}))$ above s (i.e., above the horizontal line through s) since those below s can be computed analogously. Below, with a little abuse of notation, we let $V_g(s, G_E(\mathcal{V}))$ refer to the set of its gateways on the vertical cut-lines and above s.

We follow the terminology in Section 3. Recall that s has $O(\log n)$ projection cut-lines in the vertical cut-line tree $T^v(\mathcal{V})$. Let L be the set of all projection cut-lines of s in $T^v(\mathcal{V})$. For each projection cut-line $l \in L$, let $v(l)$ be the Steiner point on l immediately above the horizontal projection $s_h(l)$ of s on l. Let $S = \{v(l) \mid l \in L\}$. By their definitions, $V_g(s, G_E(\mathcal{V}))$ is a subset of S (since each gateway of $V_g(s, G_E(\mathcal{V}))$ is on a *relevant* projection cut-line of s in $T^v(\mathcal{V})$). Hence, to compute $V_g(s, G_E(\mathcal{V}))$ and their gateway edges, it suffices to compute the set S and the weighted lengths of $\overline{ss_h(l)} \cup \overline{s_h(l)v(l)}$ for all projection cut-lines $l \in L$. Since $\overline{ss_h(l)}$ is in $\mathcal{F}$ for any projection cut-line l of s [7] (because $s \in \mathcal{F}$), it suffices to compute the weighted length of $\overline{s_h(l)v(l)}$. Below, for any line segment $\overline{ab}$, let $d_w(\overline{ab})$ denote the weighted length of $\overline{ab}$. Let $A = \{\overline{s_h(l)v(l)} \mid l \in L\}$.

We use fractional cascading [5] to obtain S in $O(\log n)$ time, with a similar approach as for Lemma 3. To compute the weighted lengths of the segments in A, we need to build another fractional cascading data structure in the preprocessing.

For every cut-line l of $T^v(\mathcal{V})$, we compute the intersections of l with all obstacle edges of $\mathcal{P}$; let $I(l)$ be the set of such intersections. Clearly, $|I(l)| = O(n)$. We sort these intersections and the Steiner points on l to obtain a sorted list $I'(l)$. For all n cut-lines of $T^v(\mathcal{V})$, this takes totally $O(n^2 \log n)$ time, because the total number of Steiner points is $O(n\sqrt{\log n}2^{\sqrt{\log n}})$ (which is $O(n^2)$) and the total number of intersections between the cut-lines and the obstacle edges is $O(n^2)$.

Consider the sorted set $I'(l)$ for any cut-line l of $T^v(\mathcal{V})$. For any two consecutive points p_1 and p_2 in $I'(l)$, the entire segment $\overline{p_1p_2}$ is either in $\mathcal{F}$ or in the same obstacle. From top to bottom in $I'(l)$, for each point $p \in I'(l)$, we compute the weighted length $d_w(\overline{pp^*})$ and associate it with p, where p^* is the highest point in $I'(l)$. Further, for each point $p \in I'(l)$, we maintain a weight w_p, defined as follows: Suppose p' is the point in $I'(l)$ immediately below p; if the interior of $\overline{pp'}$ is contained in an obstacle, then w_p is the weight of that obstacle, and $w_p = 0$ otherwise. Since $I'(l)$ is sorted, computing such information in $I'(l)$ takes $O(|I'(l)|)$ time. With such information, for any query point q on l, suppose p is the point in $I'(l)$ that is immediately above q; then we have $d_w(\overline{qp^*}) = d_w(\overline{pp^*}) + (1 + w_p) \cdot |\overline{pq}|$, where $|\overline{pq}|$ is the length of $\overline{pq}$. Hence, once we know the point p for q, $d_w(\overline{qp^*})$ can be computed in $O(1)$ time; further, for any point q' in $I'(l)$ above q, we have $d_w(\overline{qq'}) = d_w(\overline{qp^*}) - d_w(\overline{q'p^*})$, which is computed in $O(1)$ time since the value $d_w(\overline{q'p^*})$ is already stored at q'.

In the preprocessing, we build another fractional cascading data structure on $T^v(\mathcal{V})$ and the sorted lists $I'(l)$ for all cut-lines l of $T^v(\mathcal{V})$, which takes $O(n^2)$ space and $O(n^2 \log n)$ time.

For any query point s, we first use a similar approach as for Lemma 3 to compute the set S in $O(\log n)$ time. For each projection cut-line $l \in L$, let $v'(l)$ be the point in $I'(l)$ immediately above $s_h(l)$. Note that $v'(l)$ is between $v(l)$ and $s_h(l)$. We can use the above fractional cascading data structure to compute the points $v'(l)$ for all $l \in L$ in $O(\log n)$ time (since the cut-lines of L are at the nodes of a path from the root to a leaf in $T^v(\mathcal{V})$). Then for each $l \in L$, to compute $d_w(\overline{s_h(l)v(l)})$, as discussed above, we have $d_w(\overline{s_h(l)v(l)}) = d_w(\overline{s_h(l)p^*}) - d_w(\overline{v(l)p^*})$, where p^* is the highest point in $I'(l)$ and $d_w(\overline{s_h(l)p^*}) = d_w(\overline{v'(l)p^*}) + (1 + w_{v'(l)}) \cdot |\overline{s_h(l)v'(l)}|$. Since both $v(l)$ and $v'(l)$ have been computed, $d_w(\overline{s_h(l)v(l)})$ is obtained in $O(1)$ time. Hence, the weighted lengths of all segments in A are computed in $O(\log n)$ time. The lemma thus follows. □

The following theorem summarizes our algorithm for the weighted rectilinear case.

Theorem 3. *For the weighted rectilinear case, we can build a data structure of $O(n^2 \log n\, 4^{\sqrt{\log n}})$ size in $O(n^2 \log^2 n 4^{\sqrt{\log n}})$ time that can answer each query in $O(\log n)$ time (i.e., for any two query points s and t, the weighted length of a shortest s-t path can be found in $O(\log n)$ time and an actual path can be reported in additional time linear in the number of edges of the output path).*

Proof. In the preprocessing, we compute the graph $G_E(\mathcal{V})$ by Lemma 20. For each node v of $G_E(\mathcal{V})$, we compute a shortest path tree rooted at v in $G_E(\mathcal{V})$. We maintain a shortest path length table such that for any two nodes u and v in $G_E(\mathcal{V})$, the (weighted) length of the shortest path from u to v in $G_E(\mathcal{V})$ is obtained in $O(1)$ time. Computing all shortest path trees in $G_E(\mathcal{V})$ takes $O(n^2 \log n 4^{\sqrt{\log n}})$ space and $O(n^2 \log^2 n 4^{\sqrt{\log n}})$ time. We also perform the preprocessing for Lemma 21. Hence, the preprocessing takes $O(n^2 \log n 4^{\sqrt{\log n}})$ space and $O(n^2 \log^2 n 4^{\sqrt{\log n}})$ time in total.

Consider any two query points s and t. First, we use the approach in [7] to find a shortest s-t path that does not contain any obstacle vertex of $\mathcal{P}$ (if any), after a preprocessing of $O(n^2)$ time and space. Below, we focus on finding a shortest s-t path containing an obstacle vertex of $\mathcal{P}$, which must contain a gateway of s in $V_g(s, G_E(\mathcal{V}))$ and a gateway of t in $V_g(t, G_E(\mathcal{V}))$. By Lemma 21, we can compute both $V_g(s, G_E(\mathcal{V}))$ and $V_g(t, G_E(\mathcal{V}))$ in $O(\log n)$ time. Then, a shortest s-t path can be found by building a gateway graph (as discussed in Section 3) in $O(\log n)$ time since the sizes of both $V_g(s, G_E(\mathcal{V}))$ and $V_g(t, G_E(\mathcal{V}))$ are $O(\sqrt{\log n})$. As in [7], after the shortest s-t path length is computed, an actual shortest s-t path can be reported by using the shortest path trees of the nodes in $G_E(\mathcal{V})$, in time linear in the number of edges of the output path. The theorem thus follows. $\square$

Acknowledgments

D.Z. Chen's research was supported in part by NSF under Grant CCF-1217906. R. Inkulu's research was supported in part by IITG startup grant. H. Wang's research was supported in part by NSF under Grant CCF-1317143.

References

[1] S.W. Bae and Y. Okamoto. Querying two boundary points for shortest paths in a polygonal domain. *Computational Geometry: Theory and Applications*, 45:284–293, 2012.

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

[3] J.-L. De Carufel, C. Grimm, A. Maheshwari, M. Owen, and M. Smid. A note on the unsolvability of the weighted region shortest path problem. *Computational Geometry*, 47:724–727, 2014.

[4] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete and Computational Geometry*, 6:485–524, 1991.

[5] B. Chazelle and L. Guibas. Fractional cascading: I. A data structuring technique. *Algorithmica*, 1(1):133–162, 1986.

[6] D.Z. Chen, J. Hershberger, and H. Wang. Computing shortest paths amid convex pseudodisks. *SIAM Journal on Computing*, 42(3):1158–1184, 2013.

[7] D.Z. Chen, K.S. Klenk, and H.-Y.T. Tu. Shortest path queries among weighted obstacles in the rectilinear plane. *SIAM Journal on Computing*, 29(4):1223–1246, 2000.

[8] D.Z. Chen and H. Wang. A nearly optimal algorithm for finding L_1 shortest paths among polygonal obstacles in the plane. In *Proc. of the 19th European Symposium on Algorithms (ESA)*, pages 481–492, 2011.

[9] D.Z. Chen and H. Wang. Computing L_1 shortest paths among polygonal obstacles in the plane. arXiv:1202.5715v1, 2012.

[10] D.Z. Chen and H. Wang. Computing the visibility polygon of an island in a polygonal domain. In *Proc. of the 39th International Colloquium on Automata, Languages*

and Programming (ICALP), pages 218–229, 2012. Journal version published online in *Algorithmica*, 2015.

[11] D.Z. Chen and H. Wang. L_1 shortest path queries among polygonal obstacles in the plane. In *Proc. of 30th Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 293–304, 2013.

[12] D.Z. Chen and H. Wang. Computing shortest paths among curved obstacles in the plane. *ACM Transactions on Algorithms*, 11, 2015. Article No. 26.

[13] D.Z. Chen and H. Wang. Visibility and ray shooting queries in polygonal domains. *Computational Geometry: Theory and Applications*, 48:31–41, 2015.

[14] L.P. Chew. Planning the shortest path for a disc in $O(n^2 \log n)$ time. In *Proc. of the 1st Annual ACM Symposium on Computational Geometry (SoCG)*, pages 214–220, 1985.

[15] Y.-J. Chiang and J.S.B. Mitchell. Two-point Euclidean shortest path queries in the plane. In *Proc. of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 215–224, 1999.

[16] K. Clarkson, S. Kapoor, and P. Vaidya. Rectilinear shortest paths through polygonal obstacles in $O(n \log^2 n)$ time. In *Proc. of the 3rd Annual Symposium on Computational Geometry (SoCG)*, pages 251–257, 1987.

[17] K. Clarkson, S. Kapoor, and P. Vaidya. Rectilinear shortest paths through polygonal obstacles in $O(n \log^{2/3} n)$ time. Manuscript, 1988.

[18] H. Edelsbrunner, L. Guibas, and J. Stolfi. Optimal point location in a monotone subdivision. *SIAM Journal on Computing*, 15(2):317–340, 1986.

[19] H.A. ElGindy and P. Mitra. Orthogonal shortest route queries among axis parallel rectangular obstacles. *International Journal of Computational Geometry and Application*, 4:3–24, 1994.

[20] 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.

[21] L.J. Guibas, J. Hershberger, D. Leven, M. Sharir, and R.E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2(1-4):209–233, 1987.

[22] J. Hershberger. A new data structure for shortest path queries in a simple polygon. *Information Processing Letters*, 38(5):231–235, 1991.

[23] J. Hershberger and L. Guibas. An $O(n^2)$ shortest path algorithm for a non-rotating convex body. *Journal of Algorithms*, 9(1):18–46, 1988.

[24] J. Hershberger and J. Snoeyink. Computing minimum length paths of a given homotopy class. *Computational Geometry: Theory and Applications*, 4(2):63–97, 1994.

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

[26] J. Hershberger, S. Suri, and H. Yıldız. A near-optimal algorithm for shortest paths among curved obstacles in the plane. In *Proc. of the 29th Annual Symposium on Computational Geometry (SoCG)*, pages 359–368, 2013.

[27] R. Inkulu and S. Kapoor. Planar rectilinear shortest path computation using corridors. *Computational Geometry: Theory and Applications*, 42(9):873–884, 2009.

[28] R. Inkulu, S. Kapoor, and S.N. Maheshwari. A near optimal algorithm for finding Euclidean shortest path in polygonal domain. In *arXiv:1011.6481v1*, 2010.

[29] B. Joe and R.B. Simpson. Corrections to Lee's visibility polygon algorithm. *BIT*, 27:458–473, 1987.

[30] S. Kapoor and S.N. Maheshwari. Efficient algorithms for Euclidean shortest path and visibility problems with polygonal obstacles. In *Proc. of 4th Annual ACM Symposium on Computational Geometry (SoCG)*, pages 172–182, 1988.

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

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

[33] D.T. Lee. Visibility of a simple polygon. *Computer Vision, Graphics, and Image Processing*, 22(2):1983, 207–221.

[34] D.T. Lee and F.P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks*, 14(3):393–410, 1984.

[35] D.T. Lee, C.D. Yang, and T.H. Chen. Shortest rectilinear paths among weighted obstacles. *International Journal of Computational Geometry and Applications*, 1(2):109–124, 1991.

[36] J.S.B. Mitchell. An optimal algorithm for shortest rectilinear paths among obstacles. Abstracts of the *1st Canadian Conference on Computational Geometry (CCCG)*, 1989.

[37] J.S.B. Mitchell. A new algorithm for shortest paths among obstacles in the plane. *Annals of Mathematics and Artificial Intelligence*, 3(1):83–105, 1991.

[38] J.S.B. Mitchell. L_1 shortest paths among polygonal obstacles in the plane. *Algorithmica*, 8(1):55–88, 1992.

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

[40] J.S.B. Mitchell. Geometric shortest paths and network optimization, in *Handbook of Computational Geometry,* J.-R Sack and J. Urrutia (eds.), pages 633–702. Elsevier, Amsterdam, the Netherlands, 2000.

[41] J.S.B. Mitchell and C.H. Papadimitriou. The weighted region problem: Finding shortest paths through a weighted planar subdivision. *Journal of the ACM*, 38:18–73, 1991.