A Nearly Optimal Algorithm for Finding L_1 Shortest Paths among Polygonal Obstacles in the Plane*

Danny Z. Chen and Haitao Wang**

Department of Computer Science and Engineering
University of Notre Dame, Notre Dame, IN 46556, USA
{dchen, hwang6}@nd.edu

Abstract. Given a set of h pairwise disjoint polygonal obstacles of totally n vertices in the plane, we study the problem of computing an L_1 (or rectilinear) shortest path between two points avoiding the obstacles. Previously, this problem has been solved in $O(n \log n)$ time and $O(n)$ space, or alternatively in $O(n + h \log^{1.5} n)$ time and $O(n + h \log^{1.5} h)$ space. A lower bound of $\Omega(n + h \log h)$ time and $\Omega(n)$ space can be established for this problem. In this paper, we present a nearly optimal algorithm of $O(n + h \log^{1+\epsilon} h)$ time and $O(n)$ space for the problem, where $\epsilon > 0$ is an arbitrarily small constant. Specifically, after the free space is triangulated in $O(n + h \log^{1+\epsilon} h)$ time, our algorithm finds a shortest path in $O(n + h \log h)$ time and $O(n)$ space. Our algorithm is based on novel ideas and techniques, which may be of independent interest. Our algorithm can also be extended to obtain improved results for other related problems, e.g., finding shortest paths with fixed orientations, finding approximate Euclidean shortest paths, etc. In addition, our techniques yield improved results on some shortest path query problems.

1 Introduction

Computing shortest obstacle-avoiding paths in the plane is a fundamental problem in computational geometry. The Euclidean version in which the path length is measured by the Euclidean distance has been well studied (e.g., see [4, 5, 11, 14, 18, 23, 24]). In this paper, we consider the L_1 version, defined as follows. Given a set of h pairwise disjoint polygonal obstacles, $\mathcal{P} = \{P_1, P_2, \ldots, P_h\}$, of totally n vertices and two points s and t in the plane, the plane minus the interior of the obstacles is called the *free space*. Two objects are *disjoint* if they do not intersect in their interior. The L_1 *shortest path problem*, denoted by L_1-SPP, seeks a polygonal path in the free space from s to t with the minimum L_1 distance.

A closely related problem version solvable by our approach is to find shortest rectilinear paths. A *rectilinear path* is a path each of whose edges is parallel to a coordinate axis and its length is measured by the Euclidean distances or L_1 distances of its segments (they are the same for rectilinear paths). Rectilinear

* This research was supported in part by NSF under Grant CCF-0916606.
** Corresponding author.

shortest paths are used widely in VLSI design and network wire-routing applications. As shown in [7, 20–22], it is easy to convert an arbitrary polygonal path to a rectilinear path with the same L_1 length. Thus, in this paper, we focus on computing polygonal paths measured by the L_1 distance.

The L_1-SPP problem has been studied extensively (e.g., see [3, 7, 8, 20–22, 25]). In general, there are two approaches for solving this problem: Constructing a sparse "path preserving" graph (analogous to a visibility graph), or applying the continuous Dijkstra paradigm. Clarkson, Kapoor, and Vaidya [7] constructed a graph of $O(n \log n)$ nodes and $O(n \log n)$ edges such that a shortest L_1 path can be found in the graph in $O(n \log^2 n)$ time; subsequently, they gave an algorithm of $O(n \log^{1.5} n)$ time and $O(n \log^{1.5} n)$ space [8]. Based on some observations, Chen, Klenk, and Tu [3] showed that the problem was solvable in $O(n \log^{1.5} n)$ time and $O(n \log n)$ space. By using the continuous Dijkstra paradigm, Mitchell [21, 22] solved the problem in $O(n \log n)$ time and $O(n)$ space. An $\Omega(n + h \log h)$ time lower bound can be established for solving L_1-SPP (e.g., based on the results in [9]). Hence, Mitchell's algorithm is worst-case optimal. Recently, by using a corridor structure and building a smaller size path preserving graph, Inkulu and Kapoor [16] solved the problem in $O(n + h \log^{1.5} n)$ time and $O(n + h \log^{1.5} h)$ space, which is faster than Mitchell's algorithm for small h. Note that when all polygonal obstacles in $\mathcal{P}$ are convex, to our best knowledge, there is previously no better result than those mentioned above.

1.1 Our Results

We propose a new algorithm for L_1-SPP. After a triangulation of the free space is computed (say, in $O(n + h \log^{1+\epsilon} h)$ time [1] for an arbitrarily small constant $\epsilon > 0$), our algorithm finds a shortest s-t path in $O(n + h \log h)$ time and $O(n)$ space. Thus, if the triangulation can be done optimally (i.e., in $O(n + h \log h)$ time), our algorithm is also optimal. For the convex case of L_1-SPP in which all obstacles in $\mathcal{P}$ are convex, our algorithm is optimal since the triangulation can be done in $O(n + h \log h)$ time (e.g., by the approaches in [1, 15]).

As in [21, 22], we can also extend our approach to "fixed orientation metrics" [21, 22, 26], in which case a sought path is allowed to follow only a given set of orientations. For a number c of given orientations, Mitchell's algorithm finds such a shortest path in $O(cn \log n)$ time and $O(cn)$ space, and our algorithm takes $O(n + h \log^{1+\epsilon} h + c^2 h \log h)$ time and $O(n + c^2 h)$ space. Our approach also leads to an $O(n + h \log^{1+\epsilon} h + (1/\delta) h \log h)$ time algorithm for computing a δ-optimal Euclidean shortest path among polygonal obstacles for any constant $\delta > 0$. For this problem, Mitchell's algorithm [21, 22] takes $O((\sqrt{1/\delta}) n \log n)$ time, and Clarkson's algorithm [6] runs in $O((1/\delta) n \log n)$ time. The best announced exact algorithms for the Euclidean shortest path problem take $O(n \log n)$ time [14], $O(n + h \log h \log n)$ time [17], and $O(n + h \log^{1+\epsilon} h + k)$ time [5], where k is sensitive to the input and is bounded by $O(h^2)$.

Mitchell's algorithm [21, 22] further builds a shortest path map (with respect to a given source point) of $O(n)$ space in $O(n \log n)$ time such that for any query point p, the L_1 shortest path length from s to p can be reported in $O(\log n)$

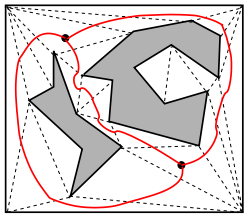

Fig. 1. Illustrating a triangulation of the free space among two obstacles and the corridors (with red solid curves). There are two junction triangles indicated by the large dots inside them, connected by three solid (red) curves. Removing the two junction triangles results in three corridors.

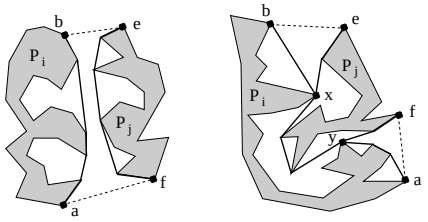

Fig. 2. Illustrating an open hourglass (left) and a closed hourglass (right) with a corridor path linking the apices x and y of the two funnels. The dashed segments are diagonals. The paths $\pi(a, b)$ and $\pi(e, f)$ are shown with thick solid curves.

time. While Mitchell's algorithm works for general polygons, this result was also the best known for the convex case of L_1-SPP. In addition, the $O(n)$ space and $O(\log n)$ query time seems optimal. As a by-product of our techniques, we can build a shortest path map whose space size is $O(h)$ (instead of $O(n)$) for the convex case of L_1-SPP in $O(n + h \log h)$ time (instead of $O(n \log n)$ time) such that any shortest path length query is answered in $O(\log h)$ time (instead of $O(\log n)$ time). This result may be of independent interest.

All our results also hold for computing L_∞ shortest paths (by rotating the plane by 45°).

1.2 An Overview of Our Approaches

We first discuss our algorithm for the convex case of L_1-SPP, which is also used as a key procedure in our algorithm for the general L_1-SPP.

In the convex case, each polygon in $\mathcal{P} = \{P_1, P_2, \ldots, P_h\}$ is convex. For each polygon $P_i \in \mathcal{P}$, we compute its *core*, denoted by $core(P_i)$, which is a simple polygon by connecting the topmost, leftmost, bottommost, and rightmost points of P_i. Let $core(\mathcal{P})$ be the set of all h cores of $\mathcal{P}$. For any two points s and t in the free space with respect to $\mathcal{P}$, we show that given a shortest s-t path avoiding all cores in $core(\mathcal{P})$, we can find in $O(n)$ time a shortest s-t path avoiding all obstacles in $\mathcal{P}$ with the same L_1 length. Based on this result, our algorithm consists of two main steps: (1) Apply Mitchell's algorithm [21, 22] on $core(\mathcal{P})$ to compute a shortest s-t path $\pi_{core}(s, t)$ avoiding the cores in $core(\mathcal{P})$, which takes $O(h \log h)$ time since each core in $core(\mathcal{P})$ has at most four vertices; (2) based on $\pi_{core}(s, t)$, compute a shortest s-t path avoiding all obstacles in $\mathcal{P}$ in $O(n)$ time. The algorithm takes overall $O(n + h \log h)$ time and $O(n)$ space.

For the general L_1-SPP, as in [16, 18], based on a triangulation of the free space, we first compute a *corridor structure*, which consists of $O(h)$ corridors and $O(h)$ junction triangles (see Fig. 1). Each corridor contains an hourglass, either open or closed (see Fig. 2). An open hourglass contains two convex chains. A closed hourglass contains two "funnels" with a *corridor path* connecting the

two apices of the two funnels. Each side of a funnel is also a convex chain. As in [16, 18], these $O(h)$ convex chains of the corridors can be used to partition the plane into a set $\mathcal{P}'$ of $O(h)$ pairwise disjoint convex polygons of totally $O(n)$ vertices such that a shortest s-t path for the original L_1-SPP is a shortest s-t path avoiding the convex polygons in $\mathcal{P}'$ and possibly utilizing some corridor paths. Further, all corridor paths are contained in the polygons of $\mathcal{P}'$. Thus, in addition to the $O(h)$ corridor paths, our L_1-SPP problem is reduced to an instance of the convex case of L_1-SPP.

But, to find a shortest s-t path for the original L_1-SPP, we cannot simply apply our algorithm for the convex case on $\mathcal{P}'$ since the corridor paths need to be considered as well. Instead, we modify Mitchell's continuous Dijkstra paradigm [21, 22] by incorporating the information of the corridor paths. Since all corridor paths are contained in the polygons of $\mathcal{P}'$, intuitively, a corridor path may provide a "shortcut" for the wavefront when it hits an endpoint of the corridor path. We derive a mechanism to handle these shortcuts. Equipping with this mechanism, the modified continuous Dijkstra paradigm can efficiently find a shortest s-t path avoiding the obstacles in $\mathcal{P}$ for L_1-SPP.

As in [21, 22], for simplicity of discussion, we assume the free space is connected (thus, a feasible s-t path always exists), and no two obstacle vertices lie on the same horizontal or vertical line. These assumptions can be removed without deteriorating the performances of the algorithms asymptotically. In the rest of this paper, unless otherwise stated, a shortest path always refers to an L_1 shortest path and a length is always in the L_1 metric.

Due to the page limit, the discussion on extending our algorithm to the fixed orientation problem is in Appendix A. We first discuss the convex case.

2 Shortest Paths among Convex Obstacles

In this section, we give our algorithm for the convex L_1-SPP, which is also used later for the general L_1-SPP in Section 3. Let $\mathcal{P}' = \{P'_1, P'_2, \ldots, P'_h\}$ be a set of h pairwise disjoint convex polygonal obstacles of totally n vertices. Given two points s and t in the free space, our algorithm finds a shortest s-t path in $O(n + h \log h)$ time and $O(n)$ space. A shortest path map (SPM for short) for handling shortest path queries can also be computed.

2.1 Notation and Observations

For each polygon $P'_i \in \mathcal{P}'$, we define its *core*, $core(P'_i)$, as the simple polygon by connecting the leftmost, topmost, rightmost, and bottommost vertices of P'_i with line segments (see Fig. 3). Note that $core(P'_i)$ is contained in P'_i and has at most four edges. Let $core(\mathcal{P}')$ be the set of the cores of all obstacles in $\mathcal{P}'$. The cores of the obstacles play a critical role in our algorithm. A key observation is that a shortest s-t path avoiding the cores in $core(\mathcal{P}')$ corresponds to a shortest s-t path avoiding the obstacles in $\mathcal{P}'$ with the same L_1 length.

To prove the above key observation, we first define some concepts. Consider an obstacle P'_i and $core(P'_i)$. For each edge $\overline{ab}$ of $core(P'_i)$ with vertices a and

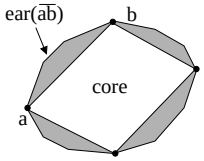

Fig. 3. Illustrating the core and ears of a convex obstacle; $ear(\overline{ab})$ is indicated.

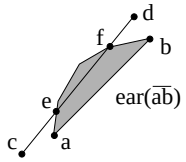

Fig. 4. The line segment $\overline{cd}$ penetrates $ear(\overline{ab})$; $\overline{cd}$ intersects the obstacle path of $ear(\overline{ab})$ at e and f.

b, if $\overline{ab}$ is not an edge of P_i', then it divides P_i' into two polygons, one of them containing $core(P_i')$; we call the one that does not contain $core(P_i')$ an *ear* of P_i' *based on* $\overline{ab}$, denoted by $ear(\overline{ab})$ (see Fig. 3). If $\overline{ab}$ is also an edge of P_i, then $ear(\overline{ab})$ is not defined. Note that $ear(\overline{ab})$ has only one edge bounding $core(P_i')$, i.e., $\overline{ab}$, which we call its *core edge*. The other edges of $ear(\overline{ab})$ are on the boundary of P_i', which we call *obstacle edges*. There are two paths between a and b along the boundary of $ear(\overline{ab})$: One path is the core edge $\overline{ab}$ and the other consists of all its obstacle edges. We call the latter path the *obstacle path* of the ear. A line segment is *positive-sloped* (resp., *negative-sloped*) if its slope is positive (resp., negative). An ear is *positive-sloped* (resp., *negative-sloped*) if its core edge is positive-sloped (resp., negative-sloped). A point p is *higher* (resp., *lower*) than another point q if the y-coordinate of p is no smaller (resp., no larger) than that of q. The next observation is self-evident.

Observation 1 *For any ear, its obstacle path is monotone in both the x- and y-coordinates. Specifically, consider an ear $ear(\overline{ab})$ and suppose the vertex a is lower than the vertex b. If $ear(\overline{ab})$ is positive-sloped, then the obstacle path from a to b is monotonically increasing in both the x- and y-coordinates; if it is negative-sloped, then the obstacle path from a to b is monotonically decreasing in the x-coordinates and monotonically increasing in the y-coordinates.*

For an ear $ear(\overline{ab})$ and a line segment $\overline{cd}$, we say that $\overline{cd}$ *penetrates* $ear(\overline{ab})$ if the following hold (see Fig. 4): (1) $\overline{cd}$ intersects the interior of $ear(\overline{ab})$, (2) neither c nor d is in the interior of $ear(\overline{ab})$, and (3) $\overline{cd}$ does not intersect the core edge $\overline{ab}$ at its interior. The next lemma will be useful later.

Lemma 1. *Suppose a line segment $\overline{cd}$ penetrates $ear(\overline{ab})$. If $\overline{cd}$ is positive-sloped (resp., negative-sloped), then $ear(\overline{ab})$ is also positive-sloped (resp., negative-sloped).*

Proof: We only show the case when $\overline{cd}$ is positive-sloped (the other case is similar).

Assume to the contrary that $ear(\overline{ab})$ is negative-sloped. Without loss of generality, we assume a is lower than b. By Observation 1, the obstacle path of $ear(\overline{ab})$ from a to b is monotonically decreasing in the x-coordinates. Thus, the rightmost point and leftmost point of $ear(\overline{ab})$ are a and b, respectively. Note that $ear(\overline{ab})$ is contained in the region between the two vertical lines passing through a and b. Since $\overline{cd}$ is positive-sloped and $\overline{ab}$ is negative-sloped, if $\overline{cd}$ intersects an interior point of $ear(\overline{ab})$, then $\overline{cd}$ must cross $\overline{ab}$ at an interior point. But since $\overline{cd}$

penetrates $ear(\overline{ab})$, $\overline{cd}$ cannot intersect any interior point of $\overline{ab}$. Hence, we have a contradiction, and the lemma follows. □

Clearly, if $\overline{cd}$ penetrates the ear $ear(\overline{ab})$, $\overline{cd}$ intersects the boundary of $ear(\overline{ab})$ at two points and both points lie on the obstacle path of $ear(\overline{ab})$ (see Fig. 4).

Lemma 2. *Suppose a line segment $\overline{cd}$ penetrates an ear $ear(\overline{ab})$. Let e and f be the two points on the obstacle path of $ear(\overline{ab})$ that $\overline{cd}$ intersects. Then the L_1 length of the line segment $\overline{ef}$ is equal to that of the portion of the obstacle path of $ear(\overline{ab})$ between e and f (see Fig. 4).*

The proof of Lemma 2 is in Appendix B. If $\overline{cd}$ penetrates $ear(\overline{ab})$, by Lemma 2, we can obtain another path from c to d by replacing $\overline{ef}$ with the portion of the obstacle path of $ear(\overline{ab})$ between e and f such that the new path has the same L_1 length as $\overline{cd}$ and the new path does not intersect the interior of $ear(\overline{ab})$.

The results in the following lemma have been proved in [21, 22].

Lemma 3. *[21, 22] There exists a shortest s-t path in the free space such that if the path makes a turn at a point p, then p must be an obstacle vertex.*

We call a shortest path that satisfies the property in Lemma 3 a *vertex-preferred shortest path*. Mitchell's algorithm [21, 22] can find a vertex-preferred shortest s-t path. Denote by $Tri(\mathcal{P}')$ a triangulation of the free space and the space inside all obstacles. Note that the free space can be triangulated in $O(n + h \log h)$ time [1, 15] and the space inside all obstacles can be triangulated in totally $O(n)$ time [2]. Hence, $Tri(\mathcal{P}')$ can be computed in $O(n + h \log h)$ time. The next lemma gives our key observation.

Lemma 4. *Given a vertex-preferred shortest s-t path that avoids the polygons in $core(\mathcal{P}')$, we can find in $O(n)$ time a shortest s-t path with the same L_1 length that avoids the obstacles in $\mathcal{P}'$.*

Proof: Consider a vertex-preferred shortest s-t path for $core(\mathcal{P}')$, denoted by $\pi_{core}(s,t)$. Suppose it makes turns at $p_1, p_2, \ldots, p_k$, ordered from s to t along the path, and each p_i is a vertex of a core in $core(\mathcal{P}')$. Let $p_0 = s$ and $p_{k+1} = t$. Then for each $i = 0, 1, \ldots, k$, the portion of $\pi_{core}(s,t)$ from p_i to p_{i+1} is the line segment $\overline{p_i p_{i+1}}$. Below, we first show that we can find a path from p_i to p_{i+1} such that it avoids the obstacles in $\mathcal{P}'$ and has the same L_1 length as $\overline{p_i p_{i+1}}$.

If $\overline{p_i p_{i+1}}$ does not intersect the interior of any obstacle in $\mathcal{P}'$, then we are done with $\overline{p_i p_{i+1}}$. Otherwise, because $\overline{p_i p_{i+1}}$ avoids $core(\mathcal{P}')$, it can intersects only the interior of some ears. Consider any such ear $ear(\overline{ab})$. Below, we prove that $\overline{p_i p_{i+1}}$ penetrates $ear(\overline{ab})$.

First, we already know that $\overline{p_i p_{i+1}}$ intersects the interior of $ear(\overline{ab})$. Second, it is obvious that neither p_i nor p_{i+1} is in the interior of $ear(\overline{ab})$. It remains to show that $\overline{p_i p_{i+1}}$ cannot intersect the core edge $\overline{ab}$ of $ear(\overline{ab})$ at the interior of $\overline{ab}$. Denote by $A' \in \mathcal{P}'$ the obstacle that contains $ear(\overline{ab})$. The interior of $\overline{ab}$ is in the interior of A'. Since $\overline{p_i p_{i+1}}$ does not intersect the interior of A', $\overline{p_i p_{i+1}}$ cannot intersect $\overline{ab}$ at its interior. Therefore, $\overline{p_i p_{i+1}}$ penetrates $ear(\overline{ab})$.

Recall that we have assumed that no two obstacle vertices lie on the same horizontal or vertical line. Since both p_i and p_{i+1} are obstacle vertices, the segment $\overline{p_i p_{i+1}}$ is either positive-sloped or negative-sloped. Without loss of generality,

assume $\overline{p_ip_{i+1}}$ is positive-sloped. By Lemma 1, $ear(\overline{ab})$ is also positive-sloped. Let e and f denote the two intersection points between $\overline{p_ip_{i+1}}$ and the obstacle path of $ear(\overline{ab})$, and $\widehat{ef}$ denote the portion of the obstacle path of $ear(\overline{ab})$ between e and f. By Lemma 2, we can replace the line segment $\overline{ef}$ ($\subseteq \overline{p_ip_{i+1}}$) by $\widehat{ef}$ to obtain a new path from p_i to p_{i+1} such that the new path has the same L_1 length as $\overline{p_ip_{i+1}}$. Further, as a portion of the obstacle path of $ear(\overline{ab})$, $\widehat{ef}$ is a boundary portion of the obstacle A' that contains $ear(\overline{ab})$, and thus $\widehat{ef}$ does not intersect the interior of any obstacle in $\mathcal{P}'$.

By processing each ear whose interior is intersected by $\overline{p_ip_{i+1}}$ as above, we find a new path from p_i to p_{i+1} such that the path has the same L_1 length as $\overline{p_ip_{i+1}}$ and the path does not intersect the interior of any obstacle in $\mathcal{P}'$.

By processing each segment $\overline{p_ip_{i+1}}$ in $\pi_{core}(s,t)$ as above for $i = 0, 1, \ldots, k$, we obtain another s-t path $\pi(s,t)$ such that the L_1 length of $\pi(s,t)$ is equal to that of $\pi_{core}(s,t)$ and $\pi(s,t)$ avoids all obstacles in $\mathcal{P}'$. We claim that $\pi(s,t)$ is a shortest s-t path avoiding the obstacles in $\mathcal{P}'$. Indeed, since each core in $core(\mathcal{P}')$ is contained in an obstacle in $\mathcal{P}'$, the length of a shortest s-t path avoiding $core(\mathcal{P}')$ cannot be longer than that of a shortest s-t path avoiding $\mathcal{P}'$. Since the length of $\pi(s,t)$ is equal to that of $\pi_{core}(s,t)$ and $\pi_{core}(s,t)$ is a shortest s-t path avoiding $core(\mathcal{P}')$, $\pi(s,t)$ is a shortest s-t path avoiding $\mathcal{P}'$.

The above discussion also provides a way to construct $\pi(s,t)$, which can be easily done in $O(n)$ time with the help of $Tri(\mathcal{P}')$. The lemma thus follows. □

2.2 The Algorithm

Our algorithm works as follows: (1) Apply Mitchell's algorithm [21, 22] on $core(\mathcal{P}')$ to find a vertex-preferred shortest s-t path avoiding the cores in $core(\mathcal{P}')$; (2) by Lemma 4, find a shortest s-t path that avoids the obstacles in $\mathcal{P}'$. The first step takes $O(h \log h)$ time and $O(h)$ space since the cores in $core(\mathcal{P}')$ have totally $O(h)$ vertices. The second step takes $O(n)$ time and $O(n)$ space.

Theorem 1. *A shortest path for the convex case of L_1-SPP can be found in $O(n + h \log h)$ time and $O(n)$ space.*

2.3 The Shortest Path Map

Mitchell's algorithm [21, 22] can also compute an SPM of size $O(n)$ for the general L_1-SPP in $O(n \log n)$ time such that a shortest path length query is answered in $O(\log n)$ time and an actual path is reported in additional time proportional to the number of turns of the path. This result was also the best known for the convex case of L_1-SPP.

By applying Mitchell's algorithm [21, 22] on the core set $core(\mathcal{P}')$ and a source point s, we can compute an SPM of size $O(h)$ in $O(h \log h)$ time, denoted by $SPM(core(\mathcal{P}'), s)$. With a planar point location data structure [10, 19], for any query point p in the free space among $\mathcal{P}'$, the length of a shortest s-p path avoiding $core(\mathcal{P}')$ can be reported in $O(\log h)$ time, which is also the length of a shortest s-p path avoiding $\mathcal{P}'$, by Lemma 4. We thus have the following result.

Theorem 2. *For the convex L_1-SPP, in $O(n + h \log h)$ time, we can build an SPM of size $O(h)$ with respect to s, such that the length of an L_1 shortest path between s and any query point in the free space can be reported in $O(\log h)$ time.*

The result in Theorem 2 is superior to Mitchell's algorithm [21, 22] in three aspects, i.e., the preprocessing time, the SPM size, and the length query time. However, with the SPM for Theorem 2, an actual shortest path avoiding $\mathcal{P}'$ between s and a query point p cannot be reported in additional time proportional to the number of turns of the path.

To process queries on actual shortest paths avoiding $\mathcal{P}'$ efficiently, in Lemma 5 below, using $SPM(core(\mathcal{P}'), s)$, we compute an SPM for $\mathcal{P}'$, denoted by $SPM(\mathcal{P}', s)$, of size $O(n)$, which can answer a shortest path length query in $O(\log h)$ time and report an actual path in time proportional to $O(\log n)$ plus the number of turns of the path. The proof of Lemma 5 is in Appendix B.

Lemma 5. *Given $SPM(core(\mathcal{P}'), s)$ for the core set $core(\mathcal{P}')$, we can compute a shortest path map $SPM(\mathcal{P}', s)$ for the obstacle set $\mathcal{P}'$ in $O(n)$ time.*

Consequently, we have the following result.

Theorem 3. *For the convex L_1-SPP, in $O(n + h \log h)$ time, we can construct an SPM of size $O(n)$ with respect to s, such that given any query point p in the free space, the length of an L_1 shortest s-p path can be reported in $O(\log h)$ time and an actual path can be found in $O(\log n + k)$ time, where k is the number of turns of the path.*

3 Shortest Paths among General Polygonal Obstacles

In this section, we consider the general L_1-SPP problem. Let $\mathcal{P} = \{P_1, P_2, \ldots, P_h\}$ be a set of h pairwise disjoint arbitrary polygonal obstacles of totally n vertices in the plane. We seek to find a shortest s-t path in the free space. After a triangulation of the free space is computed, say, in $O(n + h \log^{1+\epsilon} h)$ time [1], our algorithm runs in $O(n + h \log h)$ time and $O(n)$ space.

3.1 Preprocessing

For simplicity of discussion, we assume that all obstacles are contained in a large rectangle $\mathcal{R}$ (see Fig. 1). Denote by $\mathcal{F}$ the free space inside $\mathcal{R}$. We also view s and t as two special obstacles in $\mathcal{P}$.

Denote by $Tri(\mathcal{F})$ a triangulation of $\mathcal{F}$. Based on $Tri(\mathcal{F})$, we compute a corridor structure, which was also used in [16, 18]. This corridor structure can help reduce our problem to the convex case of L_1-SPP to some extent, as discussed below.

Let $G(\mathcal{F})$ denote the (planar) dual graph of $Tri(\mathcal{F})$, i.e., each node of $G(\mathcal{F})$ corresponds to a triangle in $Tri(\mathcal{F})$ and each edge connects two nodes of $G(\mathcal{F})$ corresponding to two triangles sharing a diagonal of $Tri(\mathcal{F})$. The degree of each node in $G(\mathcal{F})$ is at most three. Suppose there is a feasible path from s to t. As in [18], at least one node dual to a triangle incident to each of s and t is

of degree three. Based on $G(\mathcal{F})$, we compute a planar 3-regular graph, denoted by G^3 (the degree of each node in G^3 is three), possibly with loops and multi-edges, as follows. First, we remove every degree-one node from $G(\mathcal{F})$ along with its incident edge; repeat this process until no degree-one node exists. 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 exists. The resulting graph is G^3 (e.g., see Fig. 1). By Euler's formula, the resulting graph G^3 has $h+1$ faces, $2h-2$ nodes, and $3h-3$ edges [18]. Each node of G^3 corresponds to a triangle in $Tri(\mathcal{F})$, which is called a *junction triangle* (e.g., see Fig. 1). The removal of all junction triangles from G^3 results in $O(h)$ *corridors*, each of which corresponds to one edge of G^3 [18].

The boundary of a corridor C consists of four parts (see Fig. 2): (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 boundary 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 two diagonals $\overline{be}$ and $\overline{af}$ are called the *doors* of C. The corridor C is a simple polygon. Let $|C|$ denote the number of obstacle vertices on the boundary of C. In $O(|C|)$ time, we can compute the shortest path $\pi(a,b)$ (resp., $\pi(e,f)$) from a to b (resp., e to f) inside C. The region H_C bounded by $\pi(a,b), \pi(e,f)$, and the two diagonals $\overline{be}$ and $\overline{fa}$ is called an *hourglass*, which is *open* if $\pi(a,b) \cap \pi(e,f) = \emptyset$ and *closed* otherwise (see Fig. 2). 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 "funnels" and a path $\pi_C = \pi(a,b) \cap \pi(e,f)$ joining the two apices of the two funnels, called the *corridor path* of C. The two funnel apices connected by the corridor path are called the *corridor path terminals*. Each funnel side is also convex. We process each corridor as above. The total time for processing all corridors is $O(n)$.

Let Q be the union of all junction triangles and hourglasses. Then Q consists of $O(h)$ junction triangles, open hourglasses, funnels, and corridor paths. As shown in [16], there exists a shortest s-t path $\pi(s,t)$ avoiding the obstacles in $\mathcal{P}$ which is contained in Q. Consider a corridor C. If $\pi(s,t)$ contains an interior point of C, then the path $\pi(s,t)$ must intersect both doors of C; further, if the hourglass H_C of C is closed, then we claim that we can make the corridor path of C entirely contained in $\pi(s,t)$. Suppose $\pi(s,t)$ intersects the two doors of C, say, at two points p and q respectively. Then since C is a simple polygon, a Euclidean shortest path between p and q inside C, denoted by $\pi_E(p,q)$, is also an L_1 shortest path in C [13]. Note that $\pi_E(p,q)$ must contain the corridor path of C. If we replace the portion of $\pi(s,t)$ between p and q by $\pi_E(p,q)$, then we obtain a new L_1 shortest s-t path that contains the corridor path π_C. For simplicity, we still use $\pi(s,t)$ to denote the new path. In other words, $\pi(s,t)$ has the property that if $\pi(s,t)$ intersects both doors of C and the hourglass H_C is closed, then the corridor path of C is contained in $\pi(s,t)$.

Let Q' be Q minus the corridor paths. Then the boundary of Q' consists of $O(h)$ reflex vertices and $O(h)$ convex chains, implying that the complementary region $\mathcal{R} \setminus Q'$ consists of a set of polygons of totally $O(h)$ reflex vertices and

$O(h)$ convex chains. As shown in [18], the region $\mathcal{R} \setminus Q'$ can be partitioned into a set $\mathcal{P}'$ of $O(h)$ convex polygons of totally $O(n)$ vertices (e.g., by extending an angle-bisecting segment inward from each reflex vertex). In addition, for each corridor path, no portion of it lies in the free space with respect to $\mathcal{P}'$. Further, the shortest path $\pi(s,t)$ is a shortest s-t path avoiding all convex polygons in $\mathcal{P}'$ and possibly utilizing some corridor paths. The set $\mathcal{P}'$ can be easily obtained in $O(n + h \log h)$ time. Therefore, other than the corridor paths, we reduce our original L_1-SPP problem to the convex case.

3.2 The Main Algorithm

With the convex polygon set $\mathcal{P}'$, to find a shortest s-t path in $\mathcal{F}$ (i.e., the free space in $\mathcal{R}$ with respect to $\mathcal{P}$), if there is no corridor path, then we can simply apply our algorithm in Section 2. Otherwise, the situation is more complicated because the corridor paths can give possible "shortcuts" for the sought s-t path, and we must take these possible "shortcuts" into consideration while running the continuous Dijkstra paradigm. The details are given below.

First, we compute the core set $core(\mathcal{P}')$ of $\mathcal{P}'$. However, the way we construct $core(\mathcal{P}')$ here is slightly different from that in Section 2. For each convex polygon $A' \in \mathcal{P}'$, in addition to its leftmost, topmost, rightmost, and bottommost vertices, if a vertex v of A' is a corridor path terminal, then v is also kept as a vertex of the core $core(A')$. In other words, $core(A')$ is a simple (convex) polygon whose vertex set consists of the leftmost, topmost, rightmost, and bottommost vertices of A' and all corridor path terminals on A'. Since there are $O(h)$ terminal vertices, the cores in $core(\mathcal{P}')$ still have totally $O(h)$ vertices and edges. Further, the core set thus defined still has the properties discussed in Section 2 for computing shortest L_1 paths, e.g., Observation 1 and Lemmas 1, 2, and 4. Hence, by using our scheme in Section 2, we can first find a shortest s-t path avoiding the cores in $core(\mathcal{P}')$ in $O(h \log h)$ time by applying Mitchell's algorithm [21, 22], and then obtain a shortest s-t path avoiding $\mathcal{P}'$ in $O(n)$ time by Lemma 4. But, the path thus computed may not be a true shortest path in $\mathcal{F}$ since the corridor paths are not utilized. To find a true shortest path in $\mathcal{F}$, we need to modify the continuous Dijkstra paradigm when applying it on the core set $core(\mathcal{P}')$, as follows.

In Mitchell's algorithm [21, 22], when an obstacle vertex v is hit by the wavefront for the first time, it will be "permanently labeled" with a value $d(v)$, which is the length of a shortest path from s to v in the free space. The wavefront consists of many "wavelets" (each wavelet is a line segment of slope 1 or -1). The algorithm maintains a priority queue (called "event queue"), and each element in the queue is a wavelet associated with an "event point" and an "event distance", which means that the wavelet will hit the event point with the event distance. The algorithm repeatedly takes (and removes) an element from the event queue with the smallest event distance, and processes the event. After an event is processed, some new events may be added to the event queue. The algorithm stops when the point t is hit by the wavefront for the first time.

To handle the corridor paths in our problem, consider a corridor path π_C with x and y as its terminals and let l be the length of π_C. Recall that x and

y are vertices of a core in $core(\mathcal{P}')$. Consider the moment when the vertex x is permanently labeled with the distance $d(x)$. Suppose the wavefront that first hits x is from the funnel whose apex is x. Then according to our discussions above, the only way that the wavefront at x can affect a shortest s-t path is through the corridor path π_C. If y is not yet permanently labeled, then y has not been hit by the wavefront. We initiate a "pseudo-wavelet" that originates from x with the event point y and event distance $d(x)+l$, meaning that y will be hit by this pseudo-wavelet at the distance $d(x)+l$. We add the pseudo-wavelet to the event queue. If y has been permanently labeled, then the wavefront has already hit y and is currently moving along the corridor path π_C from y to x. Thus, the wavefront through x will meet the wavefront through y somewhere on the path π_C, and these two wavefronts will "die" there and never affect the free space outside the corridor. Thus, if y has been permanently labeled, then we do not need to do anything on y. In addition, at the moment when the vertex x is permanently labeled, if the wavefront that hits x is from the corridor path π_C (i.e., through y), then the wavefront will keep going to the funnel of x through x; therefore, we process this event on x as usual (i.e., as in [21, 22]), by initiating wavelets that originate from x.

Intuitively, the above treatment of corridor path terminals makes corridor paths act as possible "shortcuts" when we propagate the wavefront. The rest of the algorithm proceeds in the same way as in [21, 22] (e.g., processing the segment dragging queries). The algorithm stops when the wavefront first hits the point t, at which moment a shortest s-t path in $\mathcal{F}$ has been found.

Since there are $O(h)$ corridor paths, with the above modifications to Mitchell's algorithm as applied to $core(\mathcal{P}')$, its running time is still $O(h \log h)$. Indeed, comparing with the original continuous Dijkstra scheme [21, 22] (as applied to $core(\mathcal{P}')$), there are $O(h)$ additional events on the corridor path terminals, i.e., events corresponding to those pseudo-wavelets. To handle these additional events, we may, for example, as preprocessing, for each corridor path, associate with each its corridor path terminal x the other terminal y as well as the corridor path length l. Thus, during the algorithm, when we process the event point at x, we can find y and l immediately. In this way, each additional event is handled in $O(1)$ time in addition to adding a new event for it to the event queue. Hence, processing all events still takes $O(h \log h)$ time. Note that the shortest s-t path thus computed may penetrate some ears of $\mathcal{P}'$. As in Lemma 4, we can obtain a shortest s-t path in the free space $\mathcal{F}$ in additional $O(n)$ time. Since applying Mitchell's algorithm on $core(\mathcal{P}')$ takes $O(h)$ space, the space used in our entire algorithm is $O(n)$.

In summary, we have the following result.

Theorem 4. *A shortest path for L_1-SPP can be found in $O(n+h\log^{1+\epsilon} h)$ time (or $O(n + h \log h)$ time if a free space triangulation is given) and $O(n)$ space.*

References

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

2. B. Chazelle. Triangulating a simple polygon in linear time. *Discrete and Computational Geometry*, 6:485–524, 1991.
3. D.Z. Chen, K.S. Klenk, and H.-Y.T. Tu. Shortest path queries among weighted obstacles in the rectilinear plane. *SIAM J. on Comput.*, 29(4):1223–1246, 2000.
4. D.Z. Chen and H. Wang. Computing shortest paths amid pseudodisks. In *Proc. of the 22nd ACM-SIAM Symposium on Discrete Algorithms*, pages 309–326, 2011.
5. D.Z. Chen and H. Wang. Computing shortest paths among curved obstacles in the plane. In *arXiv:1103.3911*, 2011.
6. K. Clarkson. Approximation algorithms for shortest path motion planning. In *Proc. of the 19th ACM Symposium on Theory of Computing*, pages 56–65, 1987.
7. 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*, pages 251–257, 1987.
8. K. Clarkson, S. Kapoor, and P. Vaidya. Rectilinear shortest paths through polygonal obstacles in $O(n \log^{2/3} n)$ time. Manuscript, 1988.
9. P.J. de Rezende, D.T. Lee, and Y.F. Wu. Rectilinear shortest paths with rectangular barriers. In *Proc. of the 1st SoCG*, pages 204–213, 1985.
10. H. Edelsbrunner, L. Guibas, and J. Stolfi. Optimal point location in a monotone subdivision. *SIAM Journal on Computing*, 15(2):317–340, 1986.
11. S.K. Ghosh and D.M. Mount. An output-sensitive algorithm for computing visibility. *SIAM Journal on Computing*, 20(5):888–910, 1991.
12. L. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2:209–233, 1987.
13. J. Hershberger and J. Snoeyink. Computing minimum length paths of a given homotopy class. *Comp. Geom.: Theo. and Appl.*, 4(2):63–97, 1994.
14. J. Hershberger and S. Suri. An optimal algorithm for Euclidean shortest paths in the plane. *SIAM Journal on Computing*, 28(6):2215–2256, 1999.
15. S. Hertel and K. Mehlhorn. Fast triangulation of the plane with respect to simple polygons. *Information and Control*, 64:52–76, 1985.
16. R. Inkulu and S. Kapoor. Planar rectilinear shortest path computation using corridors. *Computational Geometry: Theory and Applications*, 42(9):873–884, 2009.
17. 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.
18. S. Kapoor, S.N. Maheshwari, and J.S.B. Mitchell. An efficient algorithm for Euclidean shortest paths among polygonal obstacles in the plane. *Discrete and Computational Geometry*, 18(4):377–383, 1997.
19. D. Kirkpatrick. Optimal search in planar subdivisions. *SIAM Journal on Computing*, 12(1):28–35, 1983.
20. R.C. Larsona and V.O. Li. Finding minimum rectilinear distance paths in the presence of barriers. *Networks*, 11:285–304, 1981.
21. J.S.B. Mitchell. An optimal algorithm for shortest rectilinear paths among obstacles. In the *1st Canadian Conference on Computational Geometry*, 1989.
22. J.S.B. Mitchell. L_1 shortest paths among polygonal obstacles in the plane. *Algorithmica*, 8(1):55–88, 1992.
23. J.S.B. Mitchell. Shortest paths among obstacles in the plane. *International Journal of Computational Geometry and Applications*, 6(3):309–332, 1996.
24. J.A. Storer and J.H. Reif. Shortest paths in the plane with polygonal obstacles. *Journal of the ACM*, 41(5):982–1012, 1994.
25. P. Widmayer. On graphs preserving rectilinear shortest paths in the presence of obstacles. *Annals of Operations Research*, 33(7):557–575, 1991.

26. P. Widmayer, Y.F. Wu, and C.K. Wong. On some distance problems in fixed orientations. *SIAM Journal on Computing*, 16(4):728–746, 1987.

Appendix

A Shortest Paths with Fixed Orientations and Approximate Euclidean Shortest Paths

This section gives some extension and applications of our shortest path algorithm for L_1-SPP.

As in [21, 22], our algorithm can be generalized to solving the C-oriented shortest path problem [26]. *A C-oriented path* is a polygonal path with each edge parallel to one of a given set C of fixed orientations. A shortest C-oriented path between two points is a C-oriented path with the minimum Euclidean distance. Rectilinear paths are a special case of this problem with two fixed orientations of 0 and $\pi/2$. Let $c = |C|$. Mitchell's algorithm [21, 22] can compute a shortest C-oriented path in $O(cn \log n)$ time and $O(cn)$ space among h pairwise disjoint polygons of totally n vertices in the plane. Similarly, our algorithm also works for this problem, as follows.

We first consider the convex case (i.e., all polygons are convex). We compute a core for each convex polygon based on the orientations in C. Note that in this case, a core has $O(c)$ vertices. Thus, we obtain a core set of totally $O(ch)$ vertices. We then apply Mitchell's algorithm for the fixed orientations of C on the core set to compute a shortest path avoiding the cores in $O(c^2 h \log h)$ time and $O(c^2 h)$ space, after which we find a shortest path avoiding the input polygons in additional $O(n)$ time as in Lemma 4. Thus, a shortest path can be found in totally $O(n + c^2 h \log h)$ time and $O(n + c^2 h)$ space. For the general case when the polygons need not be convex, the algorithm scheme is similar to our L_1 algorithm in Section 3. In summary, we have the following result.

Theorem 5. *Given a set C of orientations and a set of h pairwise disjoint polygonal obstacles of totally n vertices in the plane, we can compute a C-oriented shortest s-t path in the free space in $O(n + h \log^{1+\epsilon} h + c^2 h \log h)$ time (or $O(n + c^2 h \log h)$ time if a triangulation is given) and $O(n + c^2 h)$ space, where $c = |C|$.*

This also yields an approximation algorithm for computing a Euclidean shortest path between two points among polygonal obstacles. Since the Euclidean metric can be approximated within an accuracy of $O(1/c^2)$ if we use c equally spaced orientations, as in [21, 22], Theorem 5 leads to an algorithm that computes a path guaranteed to have a length within a factor $(1+\delta)$ of the Euclidean shortest path length, where c is chosen such that $\delta = O(1/c^2)$.

Corollary 1. *A δ-optimal Euclidean shortest path between two points among h pairwise disjoint polygons of totally n vertices in the plane can be computed in $O(n + h \log^{1+\epsilon} h + (1/\delta) h \log h)$ time (or $O(n + (1/\delta) h \log h)$ time if a triangulation is given) and $O(n + (1/\delta) h)$ space.*

B Proofs of Lemma 2 and Lemma 5

Lemma 2 *Suppose a line segment $\overline{cd}$ penetrates an ear $ear(\overline{ab})$. Let e and f be the two points on the obstacle path of $ear(\overline{ab})$ that $\overline{cd}$ intersects. Then the L_1 length of the line segment $\overline{ef}$ is equal to that of the portion of the obstacle path of $ear(\overline{ab})$ between e and f (see Fig. 4).*

Proof: Without loss of generality, suppose $\overline{cd}$ is positive-sloped and e is lower than f. By Lemma 1, $ear(\overline{ab})$ is also positive-sloped. The segment $\overline{ef}$ from e to f is monotonically increasing in both the x- and y-coordinates. Denote by $\widehat{ef}$ the portion of the obstacle path of $ear(\overline{ab})$ between e and f. Since $ear(\overline{ab})$ is positive-sloped, by Observation 1, the portion $\widehat{ef}$ from e to f is monotonically increasing in both the x- and y-coordinates. Therefore, the L_1 lengths of $\overline{ef}$ and $\widehat{ef}$ are equal. The lemma thus follows. □

Lemma 5 *Given $SPM(core(\mathcal{P}'), s)$ for the core set $core(\mathcal{P}')$, we can compute a shortest path map $SPM(\mathcal{P}', s)$ for the obstacle set $\mathcal{P}'$ in $O(n)$ time.*

Proof: Note that the polygons in $\mathcal{P}'$ are pairwise disjoint in their interior. For simplicity of discussion in this proof, we assume that any two different polygons in $\mathcal{P}'$ have disjoint interior as well as disjoint boundaries.

Consider a cell $C_{core}(r)$ with the root r in $SPM(core(\mathcal{P}'), s)$. Recall that r is always a vertex of a core in $core(\mathcal{P}')$ and all points in $C_{core}(r)$ are visible to r with respect to $core(\mathcal{P}')$ [21, 22]. In other words, for any point p in the cell $C_{core}(r)$, the line segment $\overline{rp}$ is contained in $C_{core}(r)$, and further, there exists a shortest s-p path avoiding $core(\mathcal{P}')$ that contains $\overline{rp}$.

Denote by $\mathcal{F}(\mathcal{P}')$ (resp., $\mathcal{F}(core(\mathcal{P}'))$) the free space with respect to $\mathcal{P}'$ (resp., $core(\mathcal{P}')$). Note that the cell $C_{core}(r)$ is a simple polygon in $\mathcal{F}(core(\mathcal{P}'))$. We assume that $C_{core}(r)$ contains some points in $\mathcal{F}(\mathcal{P}')$ since otherwise we do not need to consider $C_{core}(r)$.

The cell $C_{core}(r)$ may intersect some ears. In other words, certain space in $C_{core}(r)$ may be occupied by some ears. Let $C(r)$ be the subregion of $C_{core}(r)$ by removing from $C_{core}(r)$ the space occupied by all ears except their obstacle paths. Thus $C(r)$ lies in $\mathcal{F}(\mathcal{P}')$. However, for each point $p \in C(r)$, p may not be visible to r with respect to $\mathcal{P}'$. Our task here is to further decompose $C(r)$ into a set of *SPM regions* such that each such region has a root visible to all points in the region with respect to $\mathcal{P}'$; further, we need to make sure that each point q in an SPM region has a shortest path in $\mathcal{F}(\mathcal{P}')$ from s that contains the line segment connecting q and the root of the region. For this, we first show that $C(r)$ is a connected region.

To show that $C(r)$ is connected, it is sufficient to show that for any point $p \in C(r)$, there is a path in $C(r)$ that connects r and p. Consider an arbitrary point $p \in C(r)$. Since $p \in C_{core}(r)$, $\overline{rp}$ is in $C_{core}(r)$ and there is a shortest path in $\mathcal{F}(core(\mathcal{P}'))$ from s to p that contains $\overline{rp}$. If the segment $\overline{rp}$ does not intersect the interior of any ear, then we are done since $\overline{rp}$ is in $C(r)$. If $\overline{rp}$

intersects the interior of some ears, then let $ear(\overline{ab})$ be one of such ears. By the proof of Lemma 4, $\overline{rp}$ penetrates $ear(\overline{ab})$. Let e and f be the two points on the obstacle path of $ear(\overline{ab})$ that $\overline{rp}$ intersects, and $\widehat{ef}$ be the portion of the obstacle path between e and f. Note that if $\overline{rp}$ is horizontal or vertical, then it cannot penetrate $ear(\overline{ab})$ due to the monotonicity of its obstacle path by Observation 1. Without loss of generality, assume $\overline{rp}$ is positive-sloped. Then by Lemma 2, $ear(\overline{ab})$ is also positive-sloped. Recall that e and f lie on $\overline{rp}$. Without loss of generality, assume r is higher than p and f is higher than e. Then the segment $\overline{ef}$ from e to f is monotonically increasing in both the x- and y-coordinates. By Observation 1, the obstacle path portion $\widehat{ef}$ from e to f is also monotonically increasing in both the x- and y-coordinates. As in the proof of Lemma 4, for any point $q \in \widehat{ef}$, there is a shortest path in $\mathcal{F}(core(\mathcal{P}'))$ from s to q that contains $\overline{rf}$ and the portion of $\widehat{ef}$ between f and q. Since $\overline{ef}$ is on $\overline{rp}$ contained in the cell $C_{core}(r)$, by the properties of the shortest path map $SPM(\mathcal{P}', s)$ [21, 22], $\widehat{ef}$ is also contained in the cell $C_{core}(r)$. Thus, $\widehat{ef}$ is also contained in $C(r)$. If we process each ear whose interior intersects $\overline{rp}$ as above, we find a path in $C(r)$ that connects r and p; further, this path has the same L_1 length as $\overline{rp}$. Hence, $C(r)$ is a connected region.

Next, we claim that for any point $p \in C(r)$, there is a shortest path in $\mathcal{F}(\mathcal{P}')$ from s to p that contains r. Indeed, since $p \in C_{core}(r)$, there is a shortest path in $\mathcal{F}(core(\mathcal{P}'))$ from s to p that contains $\overline{rp}$; let $\pi_{core}(s, r)$ be the portion of this path between s and r. On one hand, we have shown above that there is a path from r to p in $C(r)$ with the same L_1 length as $\overline{rp}$. On the other hand, by Lemma 4, there exists a path in $\mathcal{F}(\mathcal{P}')$ from s to r with the same length as $\pi_{core}(s, r)$. Hence, a concatenation of these two paths results in a shortest path from s to p in $\mathcal{F}(\mathcal{P}')$ that contains r. Our claim thus follows.

The above claim and its proof also imply that decomposing $C(r)$ into a set of SPM regions is equivalent to computing an SPM in $C(r)$ with the vertex r as the source point, which we denote by $SPM(C(r))$. Since $C(r)$ is a connected region and $C_{core}(r)$ is a simple polygon, we claim that $C(r)$ is a (possibly degenerate) simple polygon. This is because for any ear E that intersects $C_{core}(r)$, the portion $E \cap C_{core}(r)$ lies on the boundary of the simple polygon $C_{core}(r)$; thus, removing E except its obstacle path from $C_{core}(r)$ (to form $C(r)$) changes only the boundary shape of $C_{core}(r)$ but does not change the nature of a simple polygonal region (from $C_{core}(r)$ to $C(r)$). Based on the fact that $C(r)$ is a (possibly degenerate) simple polygon, $SPM(C(r))$ can be easily computed in linear time in terms of the number of edges of $C(r)$. For example, since the Euclidean shortest path between any two points in a simple polygon is also an L_1 shortest path between the two points [13], an SPM in a simple polygon with respect to the Euclidean distance is also one with respect to the L_1 distance. Therefore, we can use a corresponding shortest path algorithm for the Euclidean case (e.g., [12]) to compute each $SPM(C(r))$ in our problem.

Note that our discussion above also implies that given $SPM(core(\mathcal{P}'), s)$, for each cell $C_{core}(r)$ with a root r, we can compute the corresponding $SPM(C(r))$

separately. Clearly, the $SPM(C(r))$'s corresponding to all cells in $SPM(core(\mathcal{P}'), s)$ constitute a shortest path map $SPM(\mathcal{P}', s)$ for $\mathcal{P}'$.

Due to the planarity of the cell regions involved, the total number of edges of all $C(r)$'s is $O(n)$. Given a triangulation $Tri(\mathcal{P}')$, all regions $C(r)$ can be obtained in totally $O(n)$ time. Computing all $SPM(C(r))$'s also takes totally $O(n)$ time. Thus, $SPM(\mathcal{P}', s)$ can be constructed in $O(n)$ time. The lemma thus follows. □