Obstacle-Avoiding Path Existence Queries in a Simple Polygon

By
Matthew Eastman

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfilment of
the requirements for the degree of
Master of Computer Science
in
Computer Science

Ottawa-Carleton Institute for Computer Science
School of Computer Science
Carleton University
Ottawa, Ontario

May 23, 2014

Abstract

A fundamental problem in computational geometry is determining whether it is possible to travel between two points in the plane while avoiding a set of obstacles. If the obstacles are known in advance, they can be preprocessed so that path-existence queries can be performed efficiently.

In this thesis, we consider a variation of the path-existence query problem where each query contains an additional set of obstacles that must be avoided. We show how to preprocess a simple polygon so that given a source and a destination within that polygon, as well as a set of pairwise disjoint convex polygon or disk obstacles, we can efficiently determine whether there is an obstacle-avoiding path between the source and the destination.

Acknowledgements

I want to thank my supervisors Anil Maheshwari and Michiel Smid for their constant support, encouragement, and invaluable advice.

I would also like to thank Vida Dujmović and Mark Lanthier for being on my thesis committee, Sasanka Roy for sharing the problem tackled by my thesis with Anil, and Paz Carmi and Carsten Grimm for helpful discussions during the 15th Korean Workshop on Computational Geometry.

I am very grateful for the funding I received from the Natural Sciences and Engineering Research Council of Canada, Carleton University, the School of Computer Science, and the Computational Geometry Lab.

My time with the Computational Geometry Lab at Carleton was thoroughly enjoyable. Prosenjit Bose, Vida Dujmović, Anil Maheshwari, Pat Morin, Michiel Smid, as well as all of the students, postdoctoral fellows, and visitors, both past and present, made the lab a welcoming and stimulating research environment.

Finally, I would like to thank my parents Gordon Eastman and Susan Eastman, and my sister Sarah Eastman, for their love and support.

Contents

Abstract **iii**

Acknowledgements **v**

1 Introduction **1**

2 Preliminaries **5**

3 Convex Polygon Obstacles **11**

3.1 Shortest paths and path existence 11

3.2 The augmented shortest path data structure 16

3.3 Obstacle-avoiding path existence queries 21

3.4 Reporting a path 26

4 Disk Obstacles **27**

4.1 The Voronoi diagram of a simple polygon 27

4.2 Hitting distance and path existence 31

4.3 The hitting distance Voronoi diagram 34

4.4 Obstacle-avoiding path existence queries 50

5 Conclusion **55**

5.1 Contributions 55

5.2 Future work 56

Bibliography **59**

List of Tables

1.1 Results on obstacle-avoiding path existence queries. Here, h denotes the number of pairwise disjoint obstacles in the query and $m = \sum_{i=1}^{h} m_i$ denotes the total number of vertices in all polygonal obstacles, where m_i is the number of vertices in the ith obstacle. 3

List of Figures

1.1 A set of disk obstacles that block all paths between s and t. 3

2.1 γ crosses α at c_1 but reflects off of α at c_2. 6
2.2 $D_2 \subseteq D_1$. 10
2.3 $D_2 \subseteq D_1 \cup D_2$ and $D_2 \setminus D_3 \subseteq \operatorname{int} D_1$. 10

3.1 $\pi(s,t)$ contains three maximal spirals $S_1 = \langle x_1, \ldots, x_6 \rangle$, $S_2 = \langle x_5, \ldots, x_{11} \rangle$, and $S_3 = \langle x_{10}, \ldots, x_{16} \rangle$. 12
3.2 (i) $\pi(s,t)$ does not contain a spiral; and (ii) $\pi(s,t)$ contains degenerate spirals. 13
3.3 ε-neighbourhoods around α_L and α_R that contain pieces of obstacle-avoiding paths between x_i and x_j. 15
3.4 The clockwise chain of $\partial\mathcal{O}$ between p_L and p_R is a curve separating s from t. 15
3.5 Constructing a polygon from a polygonal chain. 17
3.6 (i) An open hourglass; and (ii) a closed hourglass. The shaded areas show the regions of the polygon that can contain part of a shortest path that passes through both d_1 and d_2. 19
3.7 The concatenation of two hourglasses. 20
3.8 The possible types of components of $\pi(s,t) \cap \partial\mathcal{O}$. 23
3.9 The ways in which $\pi(s,t)$ can intersect an edge of $\mathcal{O}$ if $\pi(s,t)$ intersects that edge at least three times. 24

4.1 The Voronoi diagram of a set of points. 28
4.2 The Voronoi diagram of a simple polygon. 28
4.3 The images of four Voronoi edges e_1, e_2, e_3 and e_4. 30
4.4 A path between s and t in $\mathcal{V}(\mathcal{P})$. 30
4.5 If $\alpha_L \cap \partial\mathcal{P} \neq \emptyset$ then $d(c, p_L) \leq r$ and $d(c, p_R) \leq r$. 33

4.6 The hitting distance bisector $B_h(\mathbf{s}_2, \mathbf{s}_4)$ of the two sites $\mathbf{s}_2$ and $\mathbf{s}_4$ associated with the edges e_2 and e_4 of the Voronoi path shown in Figure 4.4. 36
4.7 The bisectors that make up $B_h(\mathbf{s}_2, \mathbf{s}_4)$. The shaded regions indicate which pieces of these bisectors end up forming $B_h(\mathbf{s}_2, \mathbf{s}_4)$, shown in Figure 4.6. 37
4.8 The Euclidean bisector of $I_R(e_i)$ and $I_R(e_j)$ and the hitting distance bisector of $\mathbf{s}_i$ and $\mathbf{s}_j$. 38
4.9 The Euclidean bisector of $I_R(e_i)$ and $I_R(e_j)$ and the hitting distance bisector of $\mathbf{s}_i$ and $\mathbf{s}_j$. Both bisectors contain 2-dimensional regions, shown in grey. 39
4.10 A hitting distance bisector that consists entirely of a 2-dimensional region. 40
4.11 The chosen hitting distance bisector of sites $\mathbf{s}_i$ and $\mathbf{s}_j$ in Figure 4.9 if $\mathbf{s}_i \prec \mathbf{s}_j$ (left) or if $\mathbf{s}_j \prec \mathbf{s}_i$ (right). 41
4.12 A path satisfying conditions (i) and (ii) of Lemma 4.5 when $I_R(e_i)$ is a line segment and $x \in \text{strip}(I_R(e_i))$. 42
4.13 A path satisfying conditions (i) and (ii) of Lemma 4.5 when $I_R(e_i)$ is a line segment and $x \notin \text{strip}(I_R(e_i))$. 43
4.14 A path satisfying conditions (i) and (ii) of Lemma 4.5 when $I_R(e_i)$ is a reflex vertex and $\overline{xp}$ intersects e_i. 44
4.15 A path satisfying conditions (i) and (ii) of Lemma 4.5 when $I_R(e_i)$ is a reflex vertex and $\overline{xp}$ does not intersect e_i. 45
4.16 $R_h(\mathbf{s}_i, \mathbf{s}_j)$ is unbounded when H_i contains a point p in an element of $\mathbf{s}_i$. 46
4.17 $R_h(\mathbf{s}_i, \mathbf{s}_j)$ is unbounded when $\mathbf{s}_i$ is a single-vertex site and H_i does not contain part of an element of $\mathbf{s}_i$. 47
4.18 Proof of Lemma 4.10 when $\mathbf{s}_i$ is a vertex-vertex site and H_i does not contain part of an element of $\mathbf{s}_i$. 48
4.19 The hitting distance Voronoi diagram of the sites corresponding to the edges of a path between s and t in $\mathcal{V}(\mathcal{P})$. 49
4.20 Groups $G_1, \ldots, G_6$ contain the vertices of a path between s and t in $\mathcal{V}(\mathcal{P})$. 52

List of Algorithms

4.1 Single disk obstacle-avoiding path existence query. 53

Chapter 1

Introduction

A fundamental problem in computational geometry is determining whether it is possible to travel from one point in the plane to another while avoiding a set of obstacles.

If the obstacles are polygons then we can determine whether there is an obstacle-avoiding path between two points by computing the union of the obstacles in $O(n^2)$ time and constructing a point location data structure on the resulting planar subdivision. There is an obstacle-avoiding path between the two points if and only if both points lie in the same face of this subdivision, and that face is not contained in an obstacle. The union of a set of polygons with n vertices in total has complexity $O(n^2)$, so the resulting data structure has size $O(n^2)$ and can be constructed in $O(n^2)$ time [39, 44, 8]. Checking if the query points are located in the same region of the planar subdivision can be done in $O(\log n)$ time [39, 44].

A related problem is the Euclidean shortest path problem, which asks for the shortest obstacle-avoiding path between two points in the plane. A well-studied version of this problem asks for a shortest path between two points that is contained in a given polygon. If the polygon has n vertices and is simple, meaning it does not self-intersect or contain holes, Lee and Preparata showed that the shortest path between the points that avoids the exterior of the polygon can be found in $O(n)$ time [35]. Hershberger and Suri gave an optimal $O(n \log n)$ time algorithm for polygons with holes [27]. We refer the reader to the survey on shortest paths by Mitchell [37] for more details.

Query versions of shortest path problems have received considerably less attention. In the polygonal two-point shortest path query problem, we want to preprocess a given polygon into a data structure so that the shortest path between two given points in that polygon can be found efficiently. Guibas and Hershberger showed that a simple polygon can be preprocessed in $O(n)$ time so that the shortest path can be found in $O(\log n + k)$ time, where k is the number of turns made by the shortest path [22]. This problem becomes much more difficult when the polygon has holes. If the polygon contains h holes then n is the total number of vertices on the boundary of the polygon, including the vertices of the holes. The only data structure that achieves optimal $O(\log n + k)$ query time has size $O(n^{11})$ [15]. There are several algorithms and data structures that trade query time for preprocessing time and space. Notably, Chiang and Mitchell showed that if the polygon contains h holes then it can be preprocessed in $O(n + h^5)$ time into a data structure of size $O(n + h^5)$ that supports $O(h \log n + k)$ time shortest path queries [15]. Guo et al. showed that $O(h \log n + k)$ time queries can also be achieved with an $O(n^2)$ size data structure that can be constructed in $O(n^2 \log n)$ time [23]. The recent survey by Chen [12] includes more details on shortest path query problems.

In the situations described above, the set of obstacles that must be avoided is fixed. There has been a significant amount of research conducted on finding shortest paths in dynamic environments, especially in the field of motion planning [33], but relatively little research has focused on computing data structures for path existence or shortest path *queries* in dynamic environments. The only results we know of are the dynamic point location and shortest path data structures of Chiang et al. [29] and Goodrich and Tamassia [21]. Given a planar subdivision, it is possible to construct a dynamic data structure on that subdivision that supports $O(\log^2 n)$ time point location queries and shortest path queries, that can be updated in $O(\log^2 n)$ time, where an update consists of adding an edge to the planar subdivision.

In this thesis, we consider a variation of the two-point path query problem where, in addition to a source s and a destination t, we are given a set of obstacles that must be avoided. It is possible for these obstacles to block all possible paths between s and t, as in Figure 1.1, so we are concerned with whether there is an obstacle-avoiding path between the source and destination. Specifically, we show how to preprocess a simple polygon $\mathcal{P}$ so that, given two points s and t inside $\mathcal{P}$, and a set of pairwise disjoint convex polygon or disk obstacles, we can efficiently determine whether there is an obstacle-avoiding path between s and t. A summary of our results is presented in Table 1.1.

The remainder of this thesis is organized as follows. The following chapter includes several definitions and results that will be used throughout this thesis. Importantly, we show that each obstacle can be considered separately, and that we may assume that the query points are vertices of the polygon.

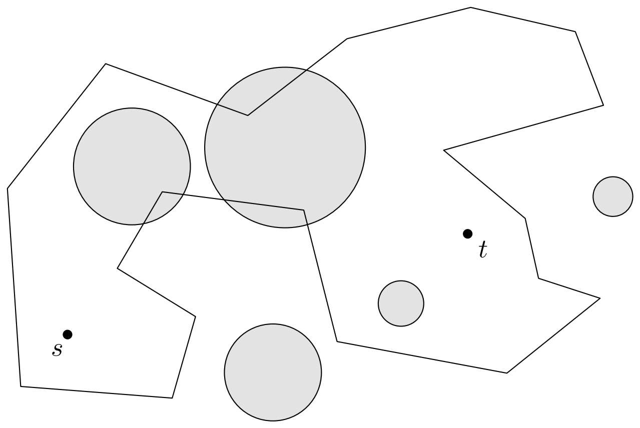


Figure 1.1: A set of disk obstacles that block all paths between s and t.

In Chapter 3, we consider obstacles that are convex polygons. We show how to augment the shortest path query data structure of Guibas and Hershberger [22] so that it can be used to answer obstacle-avoiding path existence queries.

In Chapter 4 we show how to perform obstacle-avoiding path existence queries when the obstacles are disks. We introduce a distance function that we refer to as the *hitting distance* and show that, by constructing a Voronoi diagram of a set of sites under this distance function, we can efficiently answer obstacle-avoiding path existence queries for disk obstacles.

In Chapter 5 we summarize the contributions of this thesis and suggest possible directions for future work.

Obstacles	Preprocessing	Size	Query
Pairwise disjoint convex polygons	$O(n \log n)$	$O(n \log n)$	$O(m \log^2 n)$
Pairwise disjoint disks	$O(n^2 \log n)$	$O(n^2)$	$O(h \log n)$
Pairwise disjoint disks	$O(n \log^2 n)$	$O(n \log n)$	$O(h \log^3 n)$

Table 1.1: Results on obstacle-avoiding path existence queries. Here, h denotes the number of pairwise disjoint obstacles in the query and $m = \sum_{i=1}^{h} m_i$ denotes the total number of vertices in all polygonal obstacles, where m_i is the number of vertices in the ith obstacle.

Chapter 2

Preliminaries

Let $\mathcal{P}$ be a simple polygon with n vertices. We consider $\mathcal{P}$ to be a closed and bounded (compact) subset of $\mathbb{R}^2$. Let p and q be points in $\mathcal{P}$. A *path* between p and q in $\mathcal{P}$ is a non-self-intersecting (simple) curve with endpoints p and q. Let $\mathcal{Q} = \{\mathcal{O}_1, \ldots, \mathcal{O}_h\}$ be a set of compact convex subsets of $\mathbb{R}^2$. We refer to each element $\mathcal{O}_i$, $1 \leq i \leq h$, as an *obstacle* and we say that a path γ between s and t is an *obstacle-avoiding path* if $\gamma \subseteq \mathcal{P} \setminus \bigcup_{\mathcal{O}_i \in \mathcal{Q}} \mathcal{O}_i$.

Let $\overline{xy}$ denote a line segment with endpoints x and y and let $\langle x_1, \ldots, x_k \rangle$ denote a polygonal chain with vertices $x_1, \ldots, x_k$ and edges $\overline{x_i x_{i+1}}$, $1 \leq i < k$. Let $\partial\mathcal{P}$ denote the boundary of $\mathcal{P}$ and let $\operatorname{int}\mathcal{P}$ denote its interior. Let x and y be points on $\partial\mathcal{P}$. If $\overline{xy} \subseteq \mathcal{P}$ and $\overline{xy}$ only intersects the boundary of $\mathcal{P}$ at x and y then $\overline{xy}$ is a *chord* in $\mathcal{P}$. If x and y happen to be vertices of $\mathcal{P}$ then $\overline{xy}$ is a *diagonal* in $\mathcal{P}$. Let $\langle x_1, \ldots, x_n, x_1 \rangle$ be the closed polygonal chain that forms the boundary of $\mathcal{P}$, with vertices given in clockwise order. Given two vertices x_i and x_j of $\mathcal{P}$, the directed *clockwise chain* of $\partial\mathcal{P}$ from x_i to x_j is

$$\mathcal{P}^R_{x_i x_j} = \begin{cases} \langle x_i, \ldots, x_j \rangle & \text{if } i < j, \\ \langle x_i, \ldots, x_n, x_1, \ldots, x_j \rangle & \text{if } j < i; \end{cases}$$

and the *counter-clockwise chain* is

$$\mathcal{P}^L_{x_i x_j} = \begin{cases} \langle x_i, x_{i-1}, \ldots, x_1, x_n, x_{n-1}, \ldots x_j \rangle & \text{if } i < j, \\ \langle x_i, x_{i-1}, \ldots, x_j \rangle & \text{if } j < i. \end{cases}$$

Let γ be a simple curve with endpoints x and y. If we parameterize γ by a bijection $f : [0,1] \to \gamma$ such that $f(0) = x$ and $f(1) = y$ then, given points $p, q \in \gamma$, we say that $p \prec_\gamma q$ if $f^{-1}(p) < f^{-1}(q)$.

Separating curves

Let α be a curve that is homeomorphic to either a line or a circle and let γ be a simple curve. Let R_1 and R_2 be the open regions of the plane on either side of α. Assume that $\gamma \cap \alpha \neq \emptyset$ and let c be a connected component of $\gamma \cap \alpha$. We say that γ *crosses* α at c if, for every value $\varepsilon > 0$, there exist points $x_1, x_2 \in \gamma$ in the ε-neighbourhood $U^\varepsilon = \bigcup_{x \in c} \operatorname{int} \mathcal{D}(x, \varepsilon)$ around c such that $x_1 \in R_1$ and $x_2 \in R_2$. See component c_1 in Figure 2.1. Otherwise, we say that γ *reflects* off of γ at c. See component c_2 in Figure 2.1.

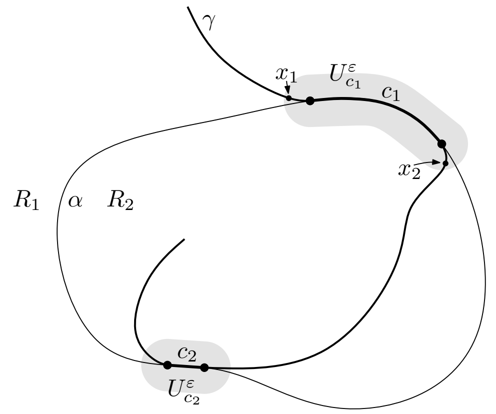


Figure 2.1: γ crosses α at c_1 but reflects off of α at c_2.

Let γ be a simple curve with endpoints x and y on the boundary of $\mathcal{P}$ such that $\gamma \subseteq P$. We say that γ *separates* s from t if γ splits $\mathcal{P}$ into two regions R_1 and R_2 with common boundary γ such that $s \in R_1$ and $t \in R_2$. If γ separates s from t then every path between s and t in $\mathcal{P}$ must cross from R_1 to R_2 by passing through γ. Let $\mathcal{O}_i$ be an obstacle in $\mathcal{Q}$. We can determine whether there is a path between s and t that avoids $\mathcal{O}_i$ by checking if there is a curve $\gamma \subseteq \mathcal{O}_i$ that separates s from t.

Lemma 2.1. *Let s and t be vertices of $\mathcal{P}$. There is an obstacle-avoiding path between s and t if and only if there is no simple curve γ separating s from t such that $\gamma \subseteq \mathcal{O}_i$.*

Proof. If there is no curve $\gamma \subseteq \mathcal{O}_i$ separating s from t then $s \notin \mathcal{O}_i$ and $t \notin \mathcal{O}_i$. Consider the overlay of $\partial\mathcal{P}$ and $\partial\mathcal{O}_i$. Both s and t must belong to the boundary of a face $F \subseteq \mathcal{P}$ in this overlay, otherwise $\mathcal{P}$ is not a simple polygon, or there is an edge in this overlay that is a curve in $\mathcal{O}_i$ separating s from t. Since the interior of $\mathcal{P}$ is path connected and the clockwise and counter-clockwise arcs of the boundary of F cannot intersect, there is a path between s and t in F that only intersects ∂F at s and t. Therefore, there is a path between s and t in $\mathcal{P} \setminus \mathcal{O}_i$.

Suppose that there is a curve $\gamma \subseteq \mathcal{O}_i$ separating s from t. Let R_1 and R_2 be the regions of $\mathcal{P}$ split by γ that contain s and t, respectively. Every path between s and t in $\mathcal{P}$ must cross from R_1 to R_2 by passing through γ, so every path between s and t intersects $\mathcal{O}_i$. Therefore, there cannot be a path between s and t in $\mathcal{P} \setminus \mathcal{O}_i$. $\square$

Observe that a closed simple curve $\gamma \subseteq \mathcal{P}$ separates s from t if and only if there exist points $x \in \gamma$ and $y \in \gamma$ such that $x \in \mathcal{P}^R_{st}$ and $y \in \mathcal{P}^L_{st}$, otherwise both s and t are contained in the same region $R \subseteq \mathcal{P}$ that results from splitting $\mathcal{P}$ along γ, which implies that there is a path between s and t in R that avoids γ.

Queries with multiple obstacles

The following result shows that each obstacle can be considered independently.

Lemma 2.2. *Let s and t be points in $\mathcal{P}$ and let $\{\mathcal{O}_1, \ldots, \mathcal{O}_h\}$ be a set of h pairwise disjoint convex obstacles. There is an obstacle-avoiding path between s and t if and only if, for each obstacle $\mathcal{O}_i$, $1 \le i \le h$, there is a path between s and t in $\mathcal{P} \setminus \mathcal{O}_i$.*

Proof. If there is a path that avoids each obstacle individually then by Lemma 2.1 there is no obstacle $\mathcal{O}_i$, $1 \le i \le h$, that contains a curve separating s from t. Since the obstacles are pairwise disjoint, there cannot be a curve separating s and t contained in $\bigcup_{1 \le i \le h} \mathcal{O}_i$ either, so there must be an obstacle-avoiding path between s and t. $\square$

Throughout the rest of this thesis, unless otherwise stated, we only consider queries that contain a single obstacle $\mathcal{O}$. If $\mathcal{O}$ is a polygon then we let m be the number of vertices in $\mathcal{O}$.

Query points are vertices

The following result shows that we only need to consider queries consisting of points that are vertices of $\mathcal{P}$.

Lemma 2.3. *Let p be a point in $\mathcal{P} \setminus \mathcal{O}$. There exists a vertex v of $\mathcal{P}$ such that $\overline{pv} \subseteq \mathcal{P} \setminus \mathcal{O}$.*

Proof. Let T be a triangulation of $\mathcal{P}$ and let $\triangle xyz$ be a triangle in T, with vertices x, y, and z, that contains p. We claim that one of $\overline{px}$, $\overline{py}$, or $\overline{pz}$ is contained in $\mathcal{P} \setminus \mathcal{O}$. Assume this is not the case. Then there exist points x', y' and z' in $\overline{px}$, $\overline{py}$, and $\overline{pz}$, respectively, such that $x', y', z' \in \mathcal{O}$. Then p is contained in the triangle $\triangle x'y'z'$, which, by the convexity of $\mathcal{O}$, is a subset of $\mathcal{O}$, implying that $p \in \mathcal{O}$, which is a contradiction. $\square$

Corollary 2.4. *Let s and t be points in $\mathcal{P}$. There exist vertices s' and t' of $\mathcal{P}$ such that there is an obstacle-avoiding path between s and t if and only if there is an obstacle-avoiding path between s' and t'.*

Proof. By Lemma 2.3, there exist vertices s' and t' such that $\overline{ss'} \subseteq \mathcal{P} \setminus \mathcal{O}$ and $\overline{tt'} \subseteq \mathcal{P} \setminus \mathcal{O}$. Let γ be an obstacle-avoiding path between s and t. Then $\overline{ss'} \cup \gamma \cup \overline{tt'}$ contains an obstacle-avoiding path between s' and t'. Using the same argument, if we have an obstacle-avoiding path δ between s' and t' then $\overline{ss'} \cup \delta \cup \overline{tt'}$ contains an obstacle-avoiding path between s and t. □

The proof of Lemma 2.3 shows how to find suitable vertices s' and t'. Given a triangulation T of $\mathcal{P}$, find a triangle $\triangle xyz$ in T that contains s and check whether $\overline{sx}$, $\overline{sy}$, or $\overline{sz}$ intersect $\mathcal{O}$. By Lemma 2.3, at least one of these segments avoids $\mathcal{O}$, so we can take s' to be the endpoint of an arbitrary segment that avoids $\mathcal{O}$. The same can be done to find a suitable vertex t'. A triangulation of $\mathcal{P}$ can be constructed in $O(n)$ time $\mathcal{P}$ [11] and a point-location data structure on that triangulation can be constructed in $O(n)$ time as well [39, 44, 8]. If $\mathcal{O}$ is a convex polygon with m vertices then we can check if a line segment intersects $\mathcal{O}$ in $O(\log m)$ time. If $\mathcal{O}$ is a disk then this check can be performed in $O(1)$ time. Throughout the rest of this thesis we only consider queries such that the source s and destination t are vertices of $\mathcal{P}$.

Properties of intersecting disks

We now prove some properties of intersecting disks that will be useful in Chapter 4. Let p_x and p_y denote the x- and y-coordinates of a point $p \in \mathbb{R}^2$, respectively, and let $\mathcal{D}(c, r)$ denote the disk of radius r centered at a point $c \in \mathbb{R}^2$. We consider a disk to be a compact subset of $\mathbb{R}^2$.

Lemma 2.5. *Let $D_1 = \mathcal{D}(c_1, r_1)$ and $D_2 = \mathcal{D}(c_2, r_2)$ be disks such that $\partial D_1 \cap \partial D_2$ consists of a single point p, and $c_2 \in \overline{c_1p}$. Then $D_2 \subseteq D_1$.*

Proof. Let q be a point in D_2. Refer to Figure 2.2. Then,

$$\begin{aligned} d(c_1, q) &\leq d(c_1, c_2) + d(c_2, q) \\ &\leq d(c_1, c_2) + d(c_2, p) \\ &= d(c_1, p) \\ &= r_1. \end{aligned}$$

Therefore, $q \in D_1$ and $D_2 \subseteq D_1$. □

Lemma 2.6. *Let $D_1 = \mathcal{D}(c_1, r_1)$ and $D_2 = \mathcal{D}(c_2, r_2)$ be disks such that $\partial D_1 \cap \partial D_2$ consists of two points p_1 and p_2. Assume that $c_{1x} < c_{2x}$ and $c_{1y} = c_{2y}$ so that $p_{1x} = p_{2x}$. Let L be the line passing through p_1 and p_2. Let q be a point in D_1 lying on or to the right of L. Then $q \in D_2$.*

Proof. We may assume that L passes through the origin, so that $p_{1x} = p_{2x} = 0$, and that c_1 and c_2 lie on the x-axis, so that $c_{1y} = c_{2y} = 0$. Let q be a point in D_1 lying to the right of L. Then

$$\begin{aligned}
d(c_1, q) &\le d(c_1, p_1) \\
d(c_1, q)^2 &\le d(c_1, p_1)^2 \\
(c_{1x} - q_x)^2 + q_y^2 &\le c_{1x}^2 + p_{1y}^2 \\
c_{1x}^2 - 2c_{1x}q_x + q_x^2 + q_y^2 &\le c_{1x}^2 + p_{1y}^2 \\
-2c_{1x}q_x + q_x^2 + q_y^2 &\le p_{1y}^2.
\end{aligned}$$

Since $c_{1x} < c_{2x}$ and $q_x \ge 0$, we have $-2c_{2x}q_x \le -2c_{1x}q_x$. Thus, $-2c_{2x}q_x + q_x^2 + q_y^2 \le p_{1y}^2$, which implies that $d(c_2, q) \le d(c_2, p_1)$ and that $q \in D_2$. □

Lemma 2.7. *Let $D_1 = \mathcal{D}(c_1, r_1)$, $D_2 = \mathcal{D}(c_2, r_2)$ and $D_3 = \mathcal{D}(c_3, r_3)$ be disks such that $c_2 \in \overline{c_1 c_3}$ and $\partial D_1 \cap \partial D_2 \cap \partial D_3$ consists of two points p_1 and p_2. Then*

(i) $D_2 \subseteq D_1 \cup D_2$; and

(ii) $D_2 \setminus D_3 \subseteq \operatorname{int} D_1$.

Proof. We may assume that c_1, c_2 and c_3 lie on a horizontal line and that $c_{1x} < c_{2x} < c_{3x}$. Let L be the line passing through p_1 and p_2. Refer to Figure 2.3.

(i) Let q be a point in D_2. By Lemma 2.6, if q lies on or to the left of L, then $q \in D_1$, and if q lies on or to the right of L, then $q \in D_3$. Therefore, $D_2 \subseteq D_1 \cup D_3$.

(ii) Let q be a point in $\partial D_1 \cap D_2$. Then q lies on or to the right of L and by Lemma 2.6, $q \in D_3$. Therefore, since $D_2 \setminus D_3$ lies strictly to the left of L, $D_2 \setminus D_3 \subseteq \operatorname{int} D_1$. □

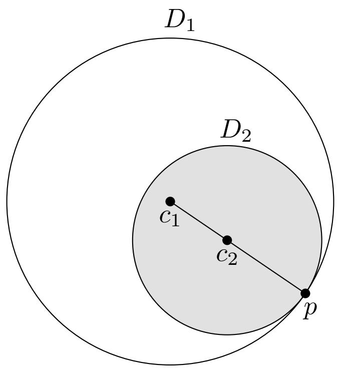


Figure 2.2: $D_2 \subseteq D_1$.

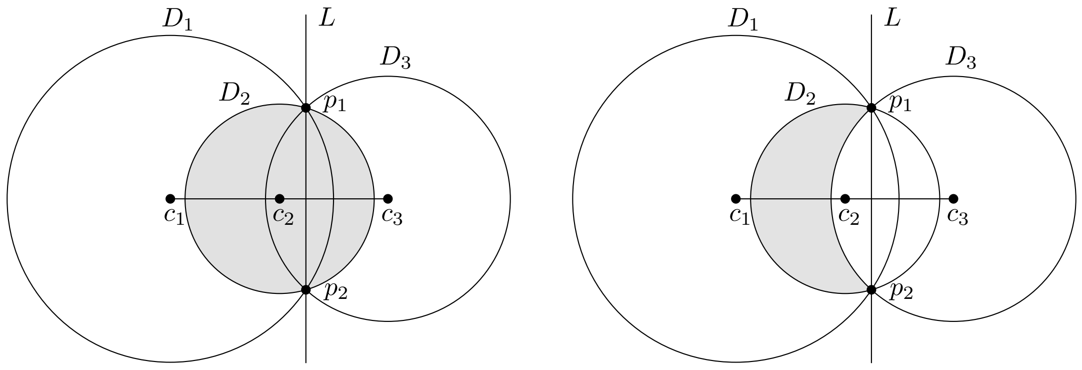


Figure 2.3: $D_2 \subseteq D_1 \cup D_2$ and $D_2 \setminus D_3 \subseteq \operatorname{int} D_1$.

Chapter 3

Convex Polygon Obstacles

In this chapter, we show how to preprocess a simple polygon $\mathcal{P}$ with n vertices into a data structure of size $O(n \log n)$ so that, given two vertices s and t of $\mathcal{P}$, and a convex polygon obstacle $\mathcal{O}$ containing m vertices, we can determine whether there is an obstacle-avoiding path between s and t in $O(m \log^2 n)$ time.

The general idea is to walk along the shortest path from s to t. Whenever we run into $\mathcal{O}$, we try to take a detour around $\mathcal{O}$ by walking along the boundary of $\mathcal{O}$ until either we meet the boundary of $\mathcal{P}$, or we rejoin the shortest path. If we can rejoin the shortest path by walking either clockwise or counter-clockwise around $\mathcal{O}$ then we continue walking towards t. If we meet the boundary of $\mathcal{P}$ in both directions then there is no obstacle-avoiding path between s and t.

In Section 3.1 we describe the shortest path between s and t and show how it can be used to perform obstacle-avoiding path existence queries. Section 3.2 shows how to augment the shortest path data structure of Guibas and Hershberger [22] in order to find the connected components of $\pi(s,t) \cap \mathcal{O}$ and Section 3.3 describes how to use these connected components to perform an obstacle-avoiding path existence query. Finally, in Section 3.4 we show how to find an obstacle-avoiding path between s and t if such a path exists.

3.1 Shortest paths and path existence

Let $\mathcal{P}$ be a simple polygon of size n. Let $\mathcal{O}$ be a convex polygon obstacle of size m and let s and t be vertices of $\mathcal{P}$. The *Euclidean shortest path* between two points p and q in $\mathcal{P}$, denoted by $\pi(p,q)$, is the path of minimum length between p and q. It is well known that $\pi(p,q)$ is a unique polygonal chain that turns only at reflex vertices of $\mathcal{P}$ [22]. The *geodesic distance* between p and q is the length of the shortest path between p and q. We treat $\pi(p, q)$ as a *directed path* from p to q so that we can define an ordering $\prec_{\pi(p,q)}$ on the points of $\pi(p, q)$ so that for any two points x and y in $\pi(p, q)$, $x \prec_{\pi(p,q)} y$ if the geodesic distance between p and x is less than the geodesic distance between p and y.

Let $\langle x_1, \ldots, x_k\rangle$ be the ordered sequence of vertices of $\mathcal{P}$ in $\pi(s, t)$, where $x_1 = s$ and $x_k = t$. We say that a subchain $S = \langle x_i, \ldots, x_j\rangle$ of $\pi(s, t)$ is a *spiral* if it makes only left, or only right, turns. Specifically, S is a left- (resp. right-) spiral if, for all $\ell < i - 2$, $x_{\ell+2}$ lies on or to the left (resp. right) of the directed line from x_ℓ through $x_{\ell+1}$. A spiral $S = \langle x_i, \ldots, x_j\rangle$ is called *maximal* if we cannot obtain a larger spiral by prepending x_{i-1} or appending x_{j+1} to S. If S is a maximal right-spiral in $\pi(s, t)$ then $x_i, x_j \in \mathcal{P}^R_{st}$ and, for all $i < \ell < j$, $x_\ell \in \mathcal{P}^L_{st}$. Symmetrically, if S is a maximal left-spiral in $\pi(s, t)$ then $x_i, x_j \in \mathcal{P}^L_{st}$ and, for all $i < \ell < j$, $x_\ell \in \mathcal{P}^R_{st}$. Figure 3.1 shows a shortest path $\pi(s, t)$ that contains three maximal spirals, $S_1 = \langle x_1, \ldots, x_6\rangle$, $S_2 = \langle x_5, \ldots, x_{11}\rangle$, and $S_3 = \langle x_{10}, \ldots, x_{16}\rangle$.

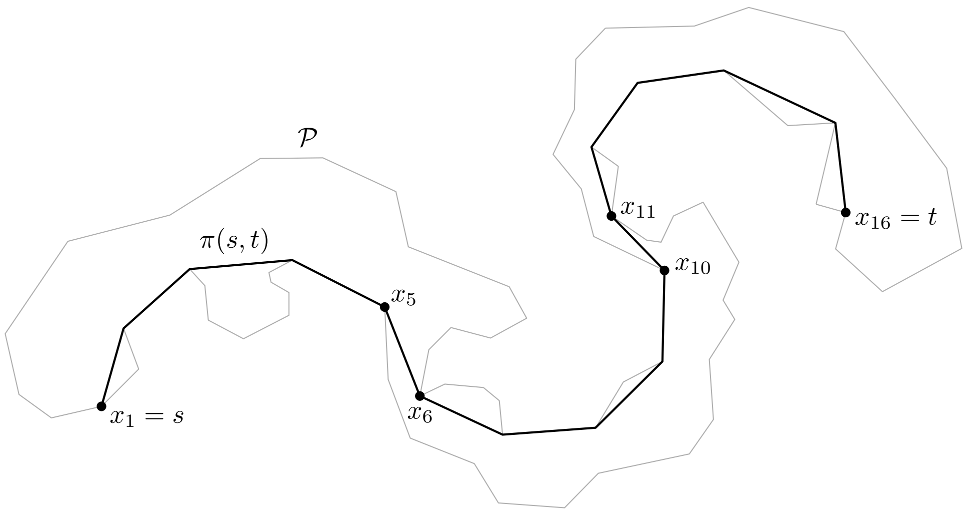
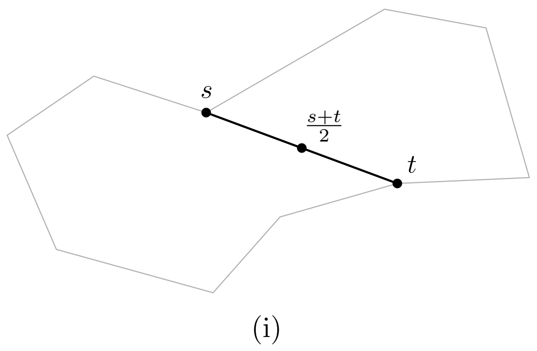


Figure 3.1: $\pi(s, t)$ contains three maximal spirals $S_1 = \langle x_1, \ldots, x_6\rangle$, $S_2 = \langle x_5, \ldots, x_{11}\rangle$, and $S_3 = \langle x_{10}, \ldots, x_{16}\rangle$.

There are two degenerate cases to consider in which $\pi(s, t)$. The first case occurs when $\pi(s, t)$ consists of a single line segment $\overline{st}$. In this case, $\pi(s, t)$ does not contain a spiral. To handle this case, we add the midpoint of s and t to $\pi(s, t)$ and say that $\pi(s, t) = \langle s, \frac{s+t}{2}, t\rangle$ is a right-spiral. See Figure 3.2(i). The second degenerate case is when $\pi(s, t)$ contains a sequence X of at least three collinear vertices of $\pi(s, t)$. If every vertex in X belongs to the same chain of $\partial\mathcal{P}$, then we say that X is a left-spiral if those vertices are on $\mathcal{P}_{st}^R$ and that X is a right-spiral if the vertices belong to $\mathcal{P}_{st}^L$. Otherwise X can be split into a set of subchains $\mathcal{S}$ such that, for each subchain $S = \langle x_i, \ldots, x_j \rangle$ in $\mathcal{S}$, $x_i, x_j \in \mathcal{P}_{st}^L$ and $x_\ell \in \mathcal{P}_{st}^R$, for all $i < \ell < j$, and we say that S is a right-spiral, or $x_i, x_j \in \mathcal{P}_{st}^R$ and $x_\ell \in \mathcal{P}_{st}^R$, for all $i < \ell < j$, and we say that S is a left-spiral. In Figure 3.2(ii), the shortest path between s and t is made up of three maximal spirals: two maximal right-spirals, $\langle x_1, x_2, x_3 \rangle$ and $\langle x_3, x_4, x_5, x_6 \rangle$; and one maximal left-spiral, $\langle x_2, x_3, x_4 \rangle$. We may now assume that $\pi(s,t)$ contains at least three vertices and at least one spiral.



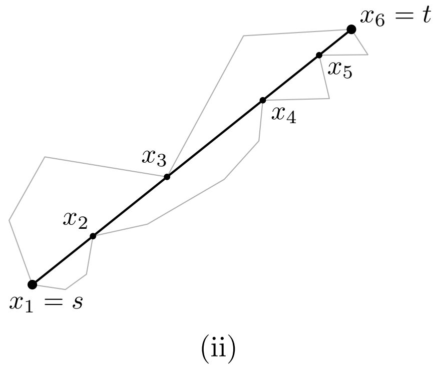


Figure 3.2: (i) $\pi(s,t)$ does not contain a spiral; and (ii) $\pi(s,t)$ contains degenerate spirals.

It is possible to decompose $\pi(s,t)$ into a sequence of $O(k)$ maximal spirals $S_1, \ldots, S_\ell$. Adjacent maximal spirals S_i and S_{i+1} overlap in exactly two vertices, meaning there is an edge of $\pi(s,t)$ shared by S_i and S_{i+1}. If S_i is a left- (resp. right-) spiral then S_{i+1} must be a right- (resp. left-) spiral, otherwise S_i and S_{i+1} are not maximal. In Figure 3.1, S_1 and S_3 are right-spirals and S_2 is a left-spiral. S_1 shares vertices x_5 and x_6 with S_2, and S_2 shares vertices x_{10} and x_{11} with S_3.

The following property of maximal spirals will be useful when we show how to determine if there is an obstacle-avoiding path.

Lemma 3.1. *Let $S = \langle x_i, \ldots, x_j \rangle$ be a maximal spiral in $\pi(s,t)$. If $x_i \in \mathcal{O}$ and $x_{i+1} \in \mathcal{O}$, or $x_{j-1} \in \mathcal{O}$ and $x_j \in \mathcal{O}$, then there cannot be an obstacle-avoiding path between s and t.*

Proof. Assume that both x_i and x_{i+1} are contained in $\mathcal{O}$ and that S is a right-spiral. If $x_i = s$ then clearly there cannot be an obstacle-avoiding path between s and t, so we may assume that $x_i \neq s$. Since S is a right-spiral, $x_i \in \mathcal{P}_{st}^R$ and $x_{i+1} \in \mathcal{P}_{st}^L$. Then $\overline{x_i x_{i+1}}$ is a chord in $\mathcal{P}$ that separates s from t. Due to the convexity of $\mathcal{O}$, $\overline{x_i x_{i+1}} \subseteq \mathcal{O}$. Therefore, there cannot be an obstacle-avoiding path between s and t.

By a similar argument, if both x_{j-1} and x_j are contained in $\mathcal{O}$ then there cannot be an obstacle-avoiding path between s and t. $\square$

Corollary 3.2. *Let c be a connected component of $\pi(s,t) \cap \mathcal{O}$. If there is no maximal spiral $S = \langle x_i, \ldots, x_j \rangle$ in $\pi(s,t)$ such that $c \subsetneq S \setminus \{x_i, x_j\}$ then there is no obstacle-avoiding path between s and t.*

Proof. If this is not the case then there must be a maximal spiral S_a in the sequence of maximal spirals that make up $\pi(s,t)$ such that c contains either the edge of $\pi(s,t)$ shared by S_a and S_{a-1} or the edge of $\pi(s,t)$ shared by S_a and S_{a+1}. Then, by Lemma 3.1, there cannot be an obstacle-avoiding path between s and t. $\square$

It immediately follows from Corollary 3.2 that, for each connected component c of $\pi(s,t) \cap \mathcal{O}$, if there is no spiral $S = \langle x_i, \ldots, x_j \rangle$ in $\pi(s,t)$ that contains c such that $x_i \notin c$ and $x_j \notin c$ then there is no obstacle-avoiding path between s and t. We now consider how to determine if there is an obstacle-avoiding path between the endpoints of a spiral S in $\pi(s,t)$ if $S \cap \mathcal{O}$ consists of a single connected component c such that c does not contain either endpoint of S.

Lemma 3.3. *Let c be a connected component of $\pi(s,t) \cap \mathcal{O}$. Suppose that there is a spiral $S = \langle x_i, \ldots, x_j \rangle$ in $\pi(s,t)$ that contains c such that $x_i \notin c$ and $x_j \notin c$. We may assume that S is the shortest spiral satisfying these constraints so that $c \cap \overline{x_i x_{i+1}} \neq \emptyset$ and $c \cap \overline{x_{j-1} x_j} \neq \emptyset$. Let y and z be the endpoints of c such that $y \prec_{\pi(s,t)} z$ and let α_R and α_L be the clockwise and counter-clockwise chains of $\partial\mathcal{O}$ with endpoints y and z, respectively. There is an obstacle-avoiding path between x_i and x_j if and only if $\alpha_R \cap \mathcal{P}^R_{st} = \emptyset$ or $\alpha_L \cap \mathcal{P}^L_{st} = \emptyset$.*

Proof. Without loss of generality, assume that S is a right-spiral. If $\alpha_L \cap \mathcal{P}^L_{st} = \emptyset$ then $y, z \in \operatorname{int} \mathcal{P}$, x_i and x_j are adjacent vertices of S, and $y, z \in \overline{x_i x_j}$. For a sufficiently small value of ε, there exists an ε-neighbourhood $U^\varepsilon_{\alpha_L} = \bigcup_{y \in \alpha_L} \operatorname{int} \mathcal{D}(y, \varepsilon)$ around α_L such that $U^\varepsilon_{\alpha_L} \subseteq \operatorname{int} \mathcal{P}$. Refer to Figure 3.3. Let y' and z' be points in $\overline{x_i y} \cap U^\varepsilon_{\alpha_L} \setminus \mathcal{O}$ and $\overline{x_j z} \cap U^\varepsilon_{\alpha_L} \setminus \mathcal{O}$, respectively. Let γ be a path between y' and z' in $U^\varepsilon_{\alpha_L} \setminus \mathcal{O}$. Then $\overline{x_i y'} \cup \gamma \cup \overline{x_j z'}$ is an obstacle-avoiding path between x_i and x_j.

A similar argument can be applied to the case where $\alpha_R \cap \mathcal{P}^R_{st} = \emptyset$. If this is the case, there exists an ε-neighbourhood $U^\varepsilon_{\alpha_R} = \bigcup_{y \in \alpha_R} \operatorname{int} \mathcal{D}(y, \varepsilon)$ around α_R such that $U^\varepsilon_{\alpha_L} \cap \mathcal{P} \setminus \mathcal{O} \subseteq \mathcal{P}$ and there is an obstacle-avoiding path between x_i to x_j that passes through $U^\varepsilon_{\alpha_L} \cap \mathcal{P} \setminus \mathcal{O} \subseteq \mathcal{P}$. Refer to Figure 3.3.

If $\alpha_R \cap \mathcal{P}^R_{st} \neq \emptyset$ and $\alpha_L \cap \mathcal{P}^L_{st} \neq \emptyset$ then when we walk along α_L from y we must first meet $\partial\mathcal{P}$ at a point p_L on $\mathcal{P}^L_{st}$, and if we walk along α_R from y, then, ignoring

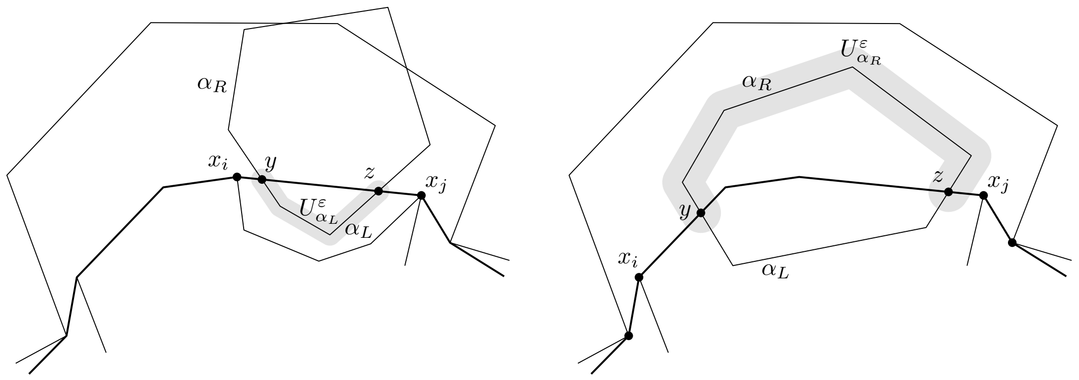


Figure 3.3: ε-neighbourhoods around α_L and α_R that contain pieces of obstacle-avoiding paths between x_i and x_j.

y if it is on $\partial\mathcal{P}$, we first meet $\partial\mathcal{P}$ at a point p_R on $\mathcal{P}^R_{st}$. This follows from the fact that $\mathcal{O}$ is convex and S makes only right turns. Refer to Figure 3.4 The union of the arc of α_R between y and p_R and the arc of the curve of α_L between y and p_L is a curve separating s from t so by Lemma 2.1 there cannot be an obstacle-avoiding path between s and t. □

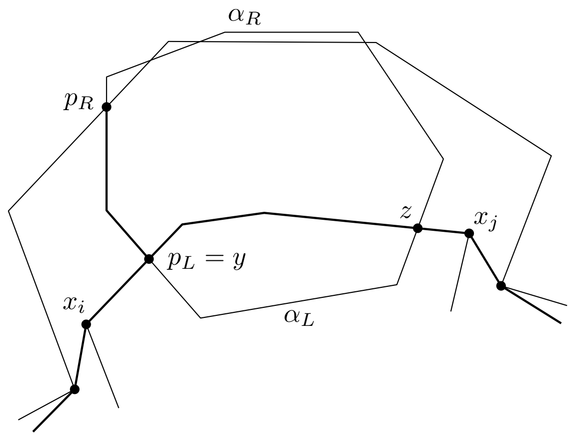


Figure 3.4: The clockwise chain of $\partial\mathcal{O}$ between p_L and p_R is a curve separating s from t.

We now describe the general idea of how to determine whether there is an obstacle-avoiding path between s and t. Walk from s towards t along $\pi(s,t)$. If we do not hit $\mathcal{O}$ then $\pi(s,t)$ is an obstacle-avoiding path between s and t. Otherwise, we meet $\mathcal{O}$ at a connected component c of $\pi(s,t) \cap \mathcal{O}$. If there is no spiral S in $\pi(s,t)$ such that $c \subseteq S$ then, by Lemma 3.2, there cannot be an obstacle-avoiding path between s and t, and we must meet $\mathcal{P}^R_{st}$ and $\mathcal{P}^L_{st}$ when walking around $\mathcal{O}$ from the endpoint of c closest to s in the clockwise and counter-clockwise directions, respectively, before reaching the other endpoint of c. If there is a spiral $S = \langle x_i, \ldots, x_j \rangle$ such that $c \subseteq S \setminus \{x_i, x_j\}$ then we may assume that S is the shortest spiral such that $x_i, x_j \notin \mathcal{O}$, and by Lemma 3.3, we can check if there is a path between x_i and x_j that avoids $\mathcal{O}$ by walking around the boundary of $\mathcal{O}$ in the clockwise and counter-clockwise directions. If we cannot walk around $\mathcal{O}$ then there cannot be an obstacle-avoiding path between s and t. Since c is the first component encountered on our walk from s to t, $\pi(s, x_i)$ does not intersect $\mathcal{O}$, so if there is an obstacle-avoiding path between x_i and x_j then there is an obstacle-avoiding path between s and x_j. If this is the case then there is an obstacle-avoiding path between s and t if and only if there is an obstacle-avoiding path between x_j and t, so by recursively solving this subproblem we can determine if there is an obstacle-avoiding path between s and t.

Thus, in order to determine whether there is an obstacle-avoiding path between s and t, we must first find the connected components of $\pi(s, t) \cap \mathcal{O}$.

3.2 The augmented shortest path data structure

In this section, we modify the shortest path data structure of Guibas and Hershberger [22] so that, given a line segment $\overline{uv}$, the I connected components of $\pi(s, t) \cap \overline{uv}$ can be found in $O(I \log^2 n)$ time. In Section 3.3 we show how this data structure can be used to find the connected components of $\pi(s, t) \cap \mathcal{O}$. We start by showing how to compute the intersections between a line segment and a polygonal chain.

Lemma 3.4. *Let $S = \langle x_1, \ldots, x_\ell \rangle$ be a polygonal chain. It is possible to preprocess S in $O(\ell)$ time into a data structure of size $O(\ell)$ so that, given a line segment $\overline{uv}$, the I intersections between S and e can be found in $O(I \log \ell)$ time.*

Proof. We can construct a polygon P_S from S as follows. Double the edges of the polygonal chain to obtain a polygon S' with boundary $\langle x_1, \ldots, x_{\ell-1}, x_\ell, x_{\ell-1}, \ldots, x_1 \rangle$. Take any axis-parallel rectangle R that contains S' in its interior and treat S' as a hole in R to obtain a polygon R'. Let x_i be a leftmost vertex of S'. Take the leftmost point r on ∂R with the same y-coordinate as x_i and carve a passage from r to x_i to obtain a polygon P_S without any holes. See Figure 3.5. We may pretend that the crack in P_S has thickness ε so that P_S is a simple polygon.

Next we construct a ray shooting data structure on S'. The ray-shooting data structure of Hershberger and Suri has size $O(\ell)$, can be constructed in $O(\ell)$ time, and allows us to perform a ray shooting query in $O(\log \ell)$ time [26]. If there are I

intersections between S and $\overline{uv}$ then we can find them in $O(I \log \ell)$ time by performing $I + 1$ ray shooting queries. □

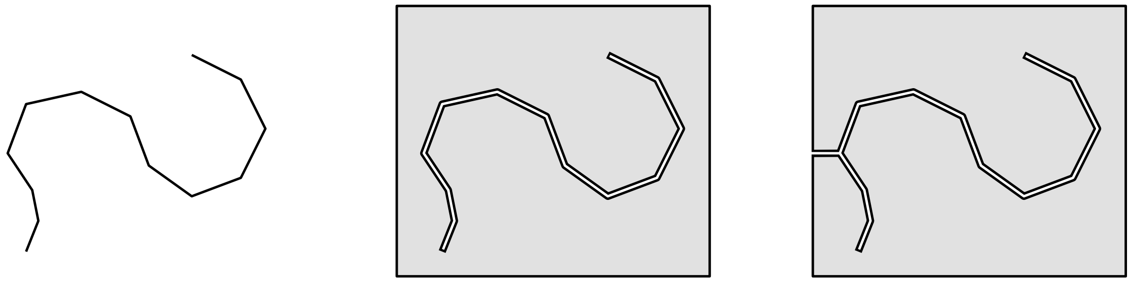


Figure 3.5: Constructing a polygon from a polygonal chain.

Let $S = \langle x_1, \ldots, x_\ell \rangle$ be a spiral. We say that S is a *semi-convex* chain if the direction of the chain changes monotonically within an open 360° interval [24]. Informally, a polygonal chain is semi-convex if when we walk around it, we turn in one direction only, and we turn less than 360° in total. We represent a semi-convex chain S as a balanced binary tree T_S using Hershberger's data structure [24], which stores the edges of S in the leaves of T_S. Each internal node v of T_S stores the common vertex of the two subchains represented by the subtrees of the children of v. Since convex chains are easier to reason about, we start by showing that a semi-convex chain can be split into two convex chains.

Lemma 3.5. *Let $S = \langle x_1, \ldots, x_\ell \rangle$ be a semi-convex chain containing at least three vertices. There is a vertex x_i, $1 < i < \ell$, such that both $\langle x_1, \ldots, x_i \rangle$ and $\langle x_i, \ldots, x_\ell \rangle$ are convex chains.*

Proof. Without loss of generality, assume that S makes only left turns. Let $i + 1$ be the first index such that $\langle x_{i+1}, x_1, x_2 \rangle$ makes a right turn instead of a left turn (if there is no such index then the chain is convex and any index $1 < j < \ell$ splits the chain into two convex chains). Then $\langle x_1, \ldots, x_i \rangle$ is a convex chain. We claim that $\langle x_i, \ldots, x_\ell \rangle$ is also convex. Assume it is not. Then $\langle x_\ell, x_i, x_{i+1} \rangle$ makes a right turn. Since x_{i+1} lies to the right of the directed line from x_1 through x_2, and also to the right of the directed line from x_ℓ through x_i, the chain must turn more than 360° in total, which contradicts the fact that S is a semi-convex chain. □

We can compute the intersections between a semi-convex chain and a line segment without having to construct a ray shooting data structure on the semi-convex chain.

Lemma 3.6. *Let $S = \langle x_1, \ldots, x_\ell \rangle$ be a semi-convex chain and let T_S be a tree representing S. Let $\overline{uv}$ be a line segment. The connected components of $S \cap \overline{uv}$ can be computed in $O(\log \ell)$ time.*

Proof. Without loss of generality, assume that S makes only left turns. By Lemma 3.5, if S is not convex then there is at least one vertex x_i that splits S into two convex chains. We can find such a vertex in $O(\log \ell)$ time using binary search, as follows. Let x_i be the vertex stored at the root of T_S. If both $\langle x_1, \ldots, x_i \rangle$ and $\langle x_i, \ldots, x_\ell \rangle$ are convex then we are done. Otherwise at least one of these chains is convex, or else S turns more than 360° and is not semi-convex, and we recursively search for a suitable split vertex in the non-convex chain.

After finding a vertex that splits S into two convex chains, we can extract the two convex subchains of S from T_S in $O(\log \ell)$ time without modifying T_S by copying the path from the root of T_S to the split vertex and modifying the copied nodes. The intersection of a convex chain and a line segment consists of zero, one, or two points, or a single line segment. We can find the connected components of the intersection between the query line segment and each of the two convex chains in $O(\log \ell)$ using binary search. Therefore, the intersections between a semi-convex chain of length ℓ and a line segment can be found in $O(\log \ell)$ time. □

We now turn our attention to the *hourglass* data structure of Guibas and Hershberger, which is used to represent all possible shortest paths between a pair of diagonals in $\mathcal{P}$ [22]. Let $d_1 = \overline{u_1 v_1}$ and $d_2 = \overline{u_2 v_2}$ be diagonals in $\mathcal{P}$ such that v_1, u_1, u_2, v_2 is a subsequence of the vertices of $\partial\mathcal{P}$ given in clockwise order. The hourglass $H(d_1, d_2)$ between d_1 and d_2 is defined as $\pi(u_1, u_2) \cup \pi(v_1, v_2)$. If $\pi(u_1, u_2) \cap \pi(v_1, v_2) = \emptyset$ then both paths are semi-convex chains, and we say that the hourglass is *open*. Otherwise the hourglass is *closed* and the shared portion of the two shortest paths is a polygonal chain with endpoints a_1 and a_2, where $a_1 \prec_{\pi(u_1,u_2)} a_2$ and $a_1 \prec_{\pi(v_1,v_2)} a_2$. The paths $\pi(u_1, a_1)$, $\pi(v_1, a_1)$, $\pi(a_2, u_2)$ and $\pi(a_2, v_2)$ are semi-convex chains. We call $\pi(a_1, a_2)$ the *string* of the hourglass. See Figure 3.6. The string of an hourglass is either a *fundamental string* consisting of two semi-convex chains linked together by a tangent line segment, or a *derived string*, which consists of two strings linked together by a fundamental string. Any piece of a fundamental or derived string may be empty.

A useful property of hourglasses is that they can be *concatenated*. Given two hourglasses $H(d_1, d_2)$ and $H(d_2, d_3)$, with d_2 separating d_1 from d_3, the hourglass $H(d_1, d_3)$ can be computed in $O(\log n)$ time [22]. If the resulting hourglass is closed, then its string is a derived string consisting of the strings of $H(d_1, d_2)$ and $H(d_2, d_3)$ linked together by a fundamental string which itself contains of a subchain of a

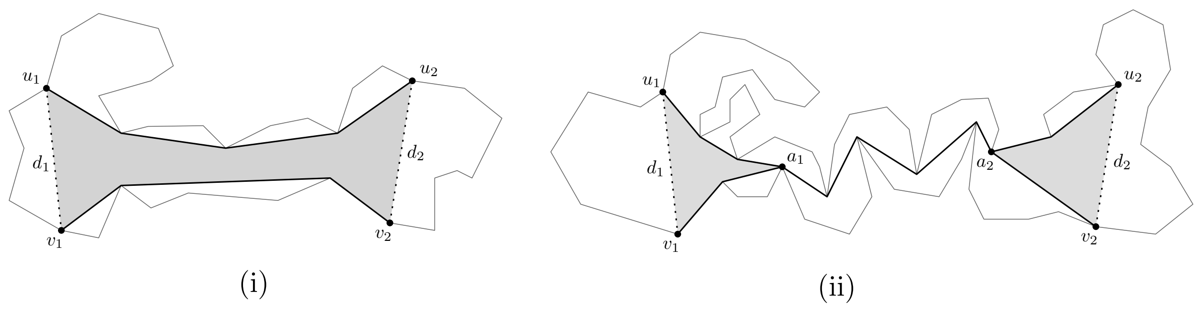


Figure 3.6: (i) An open hourglass; and (ii) a closed hourglass. The shaded areas show the regions of the polygon that can contain part of a shortest path that passes through both d_1 and d_2.

semi-convex chain in $H(d_1, d_2)$ linked to a subchain of a semi-convex chain in $H(d_2, d_3)$ by an line segment tangent to the two semi-convex chains. See Figure 3.7, or refer to [22] for more details.

Guibas and Hershberger's shortest path data structure contains a set of $O(n)$ diagonals and hourglasses such that, given points s and t in $\mathcal{P}$, a sequence of $O(\log n)$ diagonals $d_1, \ldots, d_\ell$ with the following properties can be found in $O(\log n)$ time [22].

- The triangles $\triangle su_1v_1$ and $\triangle tu_\ell v_\ell$ are contained in $\mathcal{P}$. Then, if we treat s and t as diagonals d_s and d_t, we can construct the hourglasses $H(d_s, d_1)$ and $H(d_\ell, d_t)$ in constant time by treating $\triangle su_1v_1$ and $\triangle tu_\ell v_\ell$ as open hourglasses.
- Each diagonal d_i separates s from t. Then the shortest path between s and t must pass through every diagonal in this sequence.
- Every hourglasses $H(d_i, d_{i+1})$, $1 \leq i < \ell$, was precomputed and stored in the shortest path data structure.

Then, if we concatenate the hourglasses between the diagonals in the sequence $d_s, d_1, \ldots, d_\ell, d_t$ we obtain an hourglass $H(d_s, d_t)$. If we fully expand the string of $H(d_s, d_t)$ we get a sequence of semi-convex chains and tangent line segments that is equal to $\pi(s, t)$. The hourglasses between adjacent pairs of diagonals in this sequence can be concatenated in $O(\log n)$ time each, so the string representing the shortest path between s and t can be found in $O(\log^2 n)$ time[1] [22].

For each hourglass stored in the shortest path data structure, we build a ray shooting data structure on the polygonal chain represented by the string of that

[1]Guibas and Hershberger show that the query time can be reduced to $O(\log n)$ by precomputing additional hourglasses [22], but for our purposes we only need the $O(\log^2 n)$ time data structure.

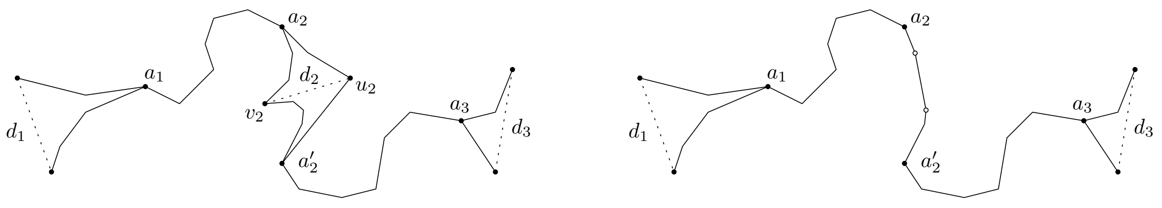


Figure 3.7: The concatenation of two hourglasses.

hourglass. Constructing a ray shooting data structure on the string of an hourglass takes linear time and space, so the size of the augmented data structure, as well as the time required to construct it, depends on the total size of the precomputed hourglasses. The pairs of diagonals for which we store a precomputed hourglasses come from the *hierarchical decomposition* of $\mathcal{P}$, which is based on the following theorem of Chazelle [10]:

Theorem 3.7 (Polygon Cutting Theorem [10])**.** *Let $\mathcal{P}$ be a simple polygon with n vertices. There exists a diagonal d in $\mathcal{P}$ that splits $\mathcal{P}$ into two sub-polygons $\mathcal{P}_1$ and $\mathcal{P}_2$, each containing at most $\frac{2}{3}n+1$ vertices.*

Applying this theorem recursively on each of the subpolygons gives us a hierarchical decomposition of $\mathcal{P}$ into a set of triangles. This decomposition can be interpreted as a binary tree T, whose internal nodes are diagonals and whose leaves are triangles that, together, form a triangulation of $\mathcal{P}$. Each internal node d is associated with the subpolygon $\mathcal{P}_d$ split by d during the decomposition. Since each diagonal splits its associated polygon into two subpolygons of approximately equal size [10], T is balanced and has height $O(\log n)$.

Each edge of $\mathcal{P}_d$ is either an edge of $\mathcal{P}$ or a diagonal that is an ancestor of d in T. Since T has height $O(\log n)$, d has $O(\log n)$ ancestors in T, meaning that each subpolygon has $O(\log n)$ edges that are are diagonals of the decomposition. For every subpolygon $\mathcal{P}_d$ in the decomposition we precompute and store an hourglass between each pair of diagonals on the boundary of $\mathcal{P}_d$. The total number of precomputed hourglasses is $O(n)$ [22]. We store a ray shooting data structure with each hourglass so the amount of space required for each hourglass is linear in the size of the hourglass.

Lemma 3.8. *A simple polygon $\mathcal{P}$ with n vertices can be preprocessed in $O(n \log n)$ time into a data structure of size $O(n \log n)$ so that, given two vertices s and t in $\mathcal{P}$, and a line segment $\overline{uv}$, the I connected components of $\pi(s,t) \cap \overline{uv}$ can be found in $O(I \log^2 n)$ time.*

Proof. We show that the augmented shortest path data structure described earlier has satisfies these properties.

We start by showing that the augmented data structure has size $O(n \log n)$ and can be computed in $O(n \log n)$ time. Let d be a diagonal in T. Let $h(d)$ denote the height of the subtree of d in T. We start by counting the number of hourglasses between d and diagonals that are descendants of d in T. If $\mathcal{P}_d$ contains ℓ vertices then $h(d) \leq 1 + \log_{3/2} \ell$, since T is balanced. We only compute an hourglass between d and one of its descendant d' if $\mathcal{P}_{d'}$ contains d on its boundary, so we compute at most two hourglasses per level. The size of the hourglass $H(d, d')$ is proportional to the size of $\mathcal{P}_{d'}$. If d' is i levels away from d' then the size of $H(d, d')$ is at most $\left(\frac{2}{3}\right)^i \ell$. The combined size of the hourglasses computed between d and its descendants in T is at most

$$\sum_{i=1}^{1+\log_{3/2} \ell} 2 \left(\tfrac{2}{3}\right)^i \ell = 4\ell - \tfrac{8}{3}.$$

Thus, the size of the augmented shortest path data structure is bounded from above by the recurrence

$$T(n) = T(\tfrac{2}{3}n + 1) + T(\tfrac{1}{3}n + 1) + 4n - \tfrac{8}{3},$$

which solves to $O(n \log n)$. We can augment an hourglass with a ray shooting data structure in linear time, so the total time required to augment every hourglass in the shortest path data structure is also $O(n \log n)$.

We can use this data structure to find a sequence of $O(\log n)$ hourglasses that represent the shortest path between s and t. After concatenating these hourglasses in $O(\log^2 n)$ time we obtain a single hourglass whose string is equivalent to $\pi(s, t)$. Partially expanding this string, we obtain a sequence consisting of $O(\log n)$ strings for which we have precomputed ray-shooting data structures, $O(\log n)$ semi-convex chains, and $O(\log n)$ tangent line segments between semi-convex chains. By Lemmas 3.4 and 3.6 we can compute the I connected components of $\pi(s, t) \cap \overline{uv}$ from this sequence in $O(I \log^2 n)$ time. □

3.3 Obstacle-avoiding path existence queries

In this section, we describe how to perform an obstacle-avoiding path existence query. The algorithm consists of two steps. The first step is to find the connected components of $\pi(s,t) \cap \mathcal{O}$. Then, for each of these components, we check if it is possible to take a detour around $\mathcal{O}$.

By Lemma 3.8, we can find the M connected components of $\pi(s,t) \cap \partial\mathcal{O}$ in $O(M \log^2 n)$ time by finding the intersections between $\pi(s,t)$ and each edge of $\partial\mathcal{O}$ separately. The connected components of $\pi(s,t) \cap \mathcal{O}$ can be found as follows. Sort the components of $\pi(s,t) \cap \partial\mathcal{O}$ by their minimum geodesic distance to s in $O(M \log M \log n)$ time. Let $c_1, \ldots, c_M$ be the sorted sequence of connected components of $\pi(s,t) \cap \mathcal{O}$. We show how to find the first connected component of $\pi(s,t) \cap \mathcal{O}$. The remaining components can be found iteratively. Referring to Figure 3.8, c_1 must be a component of type (i) or (ii), or else $s \in \mathcal{O}$. If c_1 is tangent to $\mathcal{P}$, as in case (i), then c_1 is itself a component of $\pi(s,t) \cap \mathcal{O}$. Otherwise, we are in case (ii) and $\pi(s,t)$ crosses $\partial\mathcal{O}$ to enter the interior of $\mathcal{O}$ at c_1. Then x_1 is an endpoint at the start of a connected component of $\pi(s,t) \cap \mathcal{O}$, and the next component of $\pi(s,t) \cap \mathcal{O}$ we see must belong to case (iii) or case (iv). Components belonging to case (iii), where $\pi(s,t)$ reflects off of $\partial\mathcal{O}$, cannot contain the endpoint of the first component of $\pi(s,t) \cap \mathcal{O}$, so we skip over components of type (iii). After skipping over a sequence of components of type (iii) we find a component c_i, of type (iv), such that $\pi(s,t)$ crosses $\partial\mathcal{O}$ at c_i to leave $\mathcal{O}$. Then $\pi(x_1, y_i)$ is the first connected component of $\pi(s,t) \cap \mathcal{O}$, and the next component starts at x_{i+1}, the endpoint of component c_{i+1}.

The number of connected components of $\pi(s,t) \cap \mathcal{O}$ can be linear in n. The following lemma, however, shows that we do not need to consider more than $2m$ connected components of $\pi(s,t) \cap \partial\mathcal{O}$ when performing an obstacle-avoiding path existence query.

Lemma 3.9. *Let $\overline{uv}$ be an edge of $\mathcal{O}$. If $\pi(s,t) \cap \overline{uv}$ consists of more than two connected components then there cannot be an obstacle-avoiding path between s and t.*

Proof. Let $c_1, \ldots, c_\ell$ be the components of $\pi(s,t) \cap \overline{uv}$, ordered so that $\min_{x \in c_i} d(x,u) < \min_{x \in c_{i+1}} d(x,u)$ for all $1 \le i < \ell$. Assume that $\ell \ge 3$. Consider the component c_2. Neither u nor v is visible from any point in c_2, or else c_2 would be a subset of c_1 or c_ℓ. We consider five cases based on the local structure of $\pi(s,t)$ around c_2. The cases we consider correspond to the cases shown in Figure 3.9. In these figures, the shaded regions represent the interior of $\mathcal{P}$, and the thick lines represents pieces of $\pi(s,t)$.

(i) $\pi(s,t)$ crosses $\overline{uv}$ at c_2 and c_2 consists of a single point x in the interior of $\mathcal{P}$. The ray shot from x towards u must intersect $\partial\mathcal{P}$ at a point u' before reaching u. Similarly, the ray shot from x towards v meets $\partial\mathcal{P}$ at a point v'. Then $\overline{u'v'}$ is a chord separating s from t.

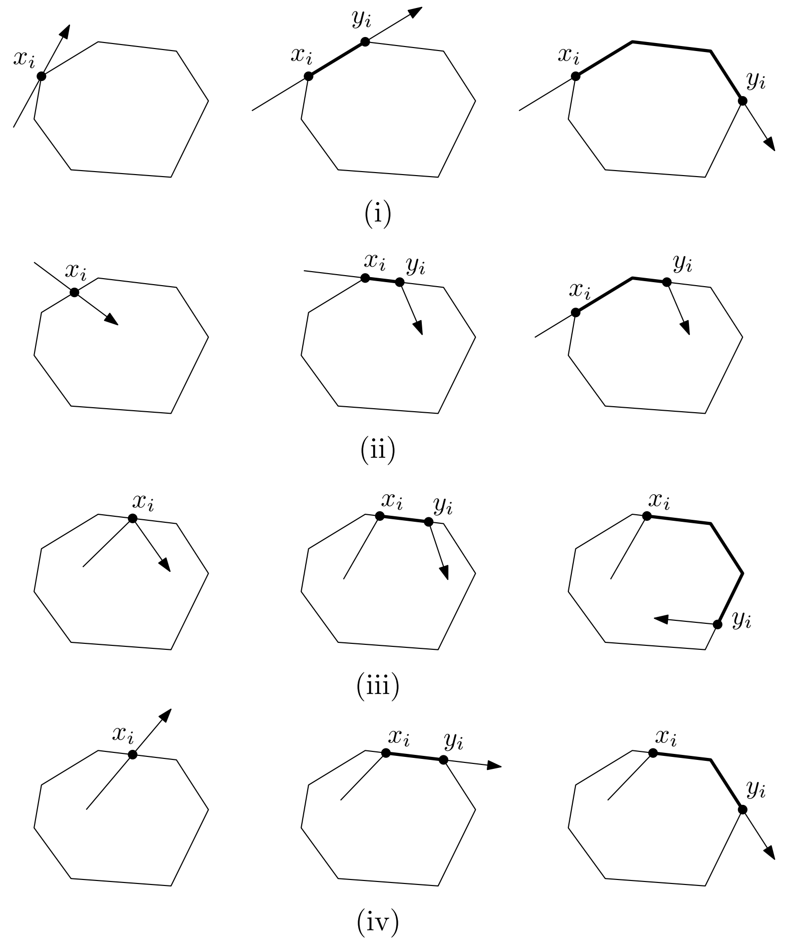


Figure 3.8: The possible types of components of $\pi(s, t) \cap \partial\mathcal{O}$.

(ii) $\pi(s, t)$ crosses $\overline{uv}$ at c_2 and c_2 consists of a single point x on the boundary of $\mathcal{P}$. Without loss of generality, assume that $x \in \mathcal{P}^L_{st}$. One of the rays shot from x towards u and from x towards v must enter into the interior of $\mathcal{O}$, or else x is a non-reflex vertex of $\mathcal{P}$ in which case $x = s$ or $x = t$ and there is no obstacle-avoiding path between s and t. Assume that the ray shot towards u enters the interior of $\mathcal{P}$. This ray first meets $\partial\mathcal{P}$ at a point u' such that $u' \in \mathcal{P}^R_{st}$. Then $\overline{xu'}$, $\overline{xu'}$ is a chord in $\mathcal{P}$ separating s from t.

(iii) $\pi(s, t)$ crosses $\overline{uv}$ at c_2 and c_2 consists of a line segment $\overline{xy}$. Both x and y must be reflex vertices of $\mathcal{P}$, and, since $\pi(s, t)$ crosses $\overline{uv}$, $\overline{xy}$ is the shared edge of two maximal spirals, which implies that x and y are on opposite chains of the boundary of $\mathcal{P}$, so $\overline{xy}$ is a chord in $\mathcal{P}$ separating s from t.

(iv) $\pi(s,t)$ reflects off $\overline{uv}$ at c_2 and c_2 consists a single point x on the boundary of $\mathcal{P}$. Then x is a reflex vertex of $\mathcal{P}$ and the rays shot from x towards u, and from x towards v, enter the interior of $\mathcal{P}$ and meet $\partial\mathcal{P}$ at points u' and v', respectively. Both $\overline{xu'}$ and $\overline{xv'}$ are chords in $\mathcal{P}$ that separate s from t.

(v) $\pi(s,t)$ reflects off $\overline{uv}$ at c_2 and c_2 consists of a line segment $\overline{uv}$. Both x and y are reflex vertices of $\mathcal{P}$, so the rays shot from x towards u, and from x towards v, enter the interior of $\mathcal{P}$ and meet $\partial\mathcal{P}$ at points u' and v', respectively. Both $\overline{xu'}$ and $\overline{xv'}$ are chords in $\mathcal{P}$ that separate s from t.

In each of the five possible cases there exist chords separating s from t that are contained in $\mathcal{O}$, implying that there cannot be an obstacle-avoiding path between s and t. □

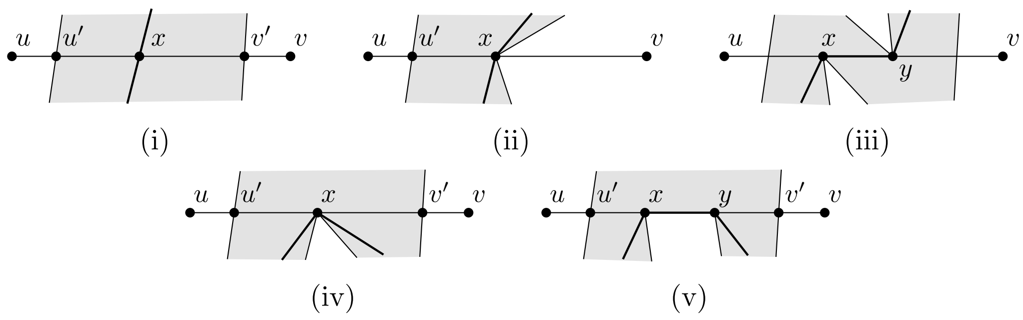


Figure 3.9: The ways in which $\pi(s,t)$ can intersect an edge of $\mathcal{O}$ if $\pi(s,t)$ intersects that edge at least three times.

Corollary 3.10. *If there is an obstacle-avoiding path between s and t then $\pi(s,t) \cap \partial\mathcal{O}$ and $\pi(s,t) \cap \mathcal{O}$ each consist of at most $2m$ connected components.*

Proof. The first claim follows directly from Lemma 3.9. Then, since each component of $\pi(s,t) \cap \mathcal{O}$ contains at least one point of $\partial\mathcal{O}$, $\pi(s,t) \cap \mathcal{O}$ consists of at most $2m$ connected components. □

Then, if we find that $\pi(s,t)$ intersects an edge of $\mathcal{O}$ more than two times, we can conclude that there cannot be an obstacle-avoiding path between s and t.

Lemma 3.11. *In $O(m \log^2 n)$ time we can either find all $O(m)$ connected components of $\pi(s,t) \cap \mathcal{O}$ or determine that there cannot be an obstacle-avoiding path between s and t.*

Proof. The M connected components of $\pi(s,t) \cap \mathcal{O}$ can be found in $O(M \log^2 n)$ time. By Corollary 3.10, when we are computing these components we can stop if we find an edge of $\mathcal{O}$ whose intersection with $\pi(s,t)$ consists of at least three connected components. So, either $M \leq 2m$, or $M > 2m$ and there must is an edge of $\mathcal{O}$ that intersects $\pi(s,t)$ at least three times and we can conclude that there cannot be an obstacle-avoiding path between s and t. $\square$

In order to check if we can walk around $\mathcal{O}$ we need to be able to check whether a line segment intersects the clockwise or counter-clockwise boundary of $\mathcal{P}$ between s and t. Let $\langle p_1, \ldots, p_n, p_1 \rangle$ be the polygonal chain that is the boundary of $\mathcal{P}$, given in clockwise order starting from an arbitrary vertex p_1. Construct a *range tree* T on the edges of $\partial\mathcal{O}$ [8]. The edges of $\partial\mathcal{O}$ are stored in the leaves of T and each internal node v contains the vertex of $\mathcal{O}$ that is common to the two subchains represented by the subtree of v. For each internal vertex v in T we construct and store a ray-shooting data structure on the subchain represented by the subtree rooted at v. This range tree can be constructed in $O(n \log n)$ time and allows us to determine whether a given line segment intersects the clockwise or counterclockwise chain between two given points s and t in $O(\log^2 n)$ time [8].

We now describe how to perform an obstacle-avoiding path existence query. Compute the connected components of $\pi(s,t) \cap \mathcal{O}$, stopping if there is an edge of $\mathcal{O}$ whose intersection with $\pi(s,t)$ consists of more than two connected components. If there is such an edge, then report that there is no obstacle-avoiding path between s and t. By Lemma 3.11, this step can be done in $O(m \log^2 n)$ time. For each connected component c in $\pi(s,t) \cap \mathcal{O}$ with endpoints $y \prec_{\pi(s,t)} z$, let α_R and α_L be the clockwise and counter-clockwise chains of $\partial\mathcal{O}$ between y and z, respectively. For each edge $\overline{uv}$ in α_R check if $\overline{uv}$ intersects $\mathcal{P}^R_{st}$. This can be done in $O(\log^2 n)$ time using the range tree T described earlier. Repeat this for each edge in α_L, checking if these edges intersect $\mathcal{P}^L_{st}$. If an edge of α_R intersected $\mathcal{P}^R_{st}$ and an edge of α_L intersected $\mathcal{P}^L_{st}$ then, by Lemma 3.3, there is no obstacle-avoiding path between s and t. This step takes $O(m \log^2 n)$ time in total, since we check if each edge of $\mathcal{O}$ intersects $\partial\mathcal{O}$ at most three times in total due to Lemma 3.9. If we are able to walk around each connected component, then there is an obstacle-avoiding path between s and t. We conclude with the following theorem.

Theorem 3.12. *A simple polygon $\mathcal{P}$ with n vertices can be preprocessed in $O(n \log n)$ time into a data structure of size $O(n \log n)$ so that, given two vertices s and t in $\mathcal{P}$, and a convex polygon obstacle $\mathcal{O}$ containing m vertices, it is possible to determine whether there is an obstacle-avoiding path between s and t in $O(m \log^2 n)$ time.*

3.4 Reporting a path

In this section, we show how to find an obstacle-avoiding path between s and t, if such a path exists.

Let $\mathcal{Q} = \{\mathcal{O}_1, \ldots, \mathcal{O}_h\}$ be the set of polygonal obstacles. We start by showing how to find a path between s and t that is contained in the closure of $\mathcal{P} \setminus \mathcal{Q}$, that is, a path between s and t that avoids the interiors of the obstacles but may intersect the boundaries of the obstacles.

Let $c_1, \ldots, c_\ell$ be the connected components of $\pi(s,t) \cap \mathcal{Q}$ ordered by their minimum geodesic distance to s. For each component c_i, let x_i and y_i be the endpoints of c_i such that $x_i \prec_{\pi(s,t)} y_i$. If there is an obstacle-avoiding path between s and t, then, by Lemma 3.3, either the clockwise or counter-clockwise chain of the obstacle that contains c_i is contained in the interior of $\mathcal{P}$, with the possible exception of the endponits x_i and y_i which can lie on the boundary of $\mathcal{P}$. Let α_i be such a chain. Then,

$$\pi(s, x_1) \cup \alpha_1 \cup \pi(y_1, x_2) \cup \ldots \cup \pi(y_{\ell-1}, x_\ell) \cup \alpha_\ell \cup \pi(y_\ell, t)$$

is a path between s and t that is contained in $\mathcal{P} \setminus \mathcal{Q}$.

Lemma 3.13. *If there is an obstacle-avoiding path between s and t then a path between s and t that is contained in $\mathcal{P} \setminus \mathcal{Q}$ can be reported in $O(m \log^2 n + k)$ time, where k is the number of turns made by $\pi(s,t)$.*

Proof. By Lemma 3.10,

$$\pi(s, x_1) \cup \alpha_1 \cup \pi(y_1, x_2) \cup \ldots \cup \pi(y_{\ell-1}, x_\ell) \cup \alpha_\ell \cup \pi(y_\ell, t)$$

makes at most $2k + 2m$ turns. Therefore, using the path existence algorithm given in Section 3.3 as well as the shortest path data structure of Guibas and Hershberger [22], we can find and report the components of this path in $O(m \log^2 n + k)$ time. $\square$

If the minimum distance between any two obstacles is $\zeta > 0$, then it is possible to find an obstacle-avoiding path between s and t that does not touch the boundaries of the obtacles. For each obstacle $\mathcal{O}_i$, construct an offset polygon $\mathcal{O}_i^\star$ such that for each point $p^\star \in \partial\mathcal{O}_i^\star$ with corresponding point $p \in \partial\mathcal{O}_i$, $0 < d(p^\star, p) < \frac{\zeta}{2}$. Then, by Lemma 3.13, we can find a path between s and t in $\mathcal{P} \setminus \bigcup_{1 \le i \le h} \mathcal{O}_i^\star$ in $O(m \log^2 n + k)$ time. This path is an obstacle-avoiding path.

Corollary 3.14. *If there is an obstacle-avoiding path between s and t, and the minimum distance $\zeta > 0$ between all pairs of obstacles is known, then an obstacle-avoiding path between s and t can be reported in $O(m \log^2 n + k)$ time, where k is the number of turns made by $\pi(s,t)$.*

Chapter 4

Disk Obstacles

In this chapter, we show how to preprocess a simple polygon $\mathcal{P}$ of size n into a data structure of size $O(n \log n)$ in $O(n \log^2 n)$ time so that given two vertices s and t of $\mathcal{P}$, and a disk obstacle $\mathcal{O}$, we can determine whether there exists an obstacle-avoiding path between s and t in $O(\log^3 n)$ time.

In Section 4.1 we describe the structure of the Voronoi diagram of $\mathcal{P}$, which will be used throughout this chapter. In Section 4.2 we introduce a distance function we call the *hitting distance* and show how it can be used to determine whether there is an obstacle-avoiding path between s and t. Section 4.3 introduces a Voronoi diagram based on the hitting distance and in Section 4.4 we show how to use this diagram to answer obstacle-avoiding path existence queries.

4.1 The Voronoi diagram of a simple polygon

The *Voronoi diagram* of a set of *Voronoi sites* $\mathcal{S}$, denoted by $\mathcal{V}(\mathcal{S})$, is a division of the plane into a set of *Voronoi regions.* Each region is associated with a site $\mathbf{s} \in \mathcal{S}$ and contains all points in $\mathbb{R}^2$ that are closer to $\mathbf{s}$ than to any other site in $\mathcal{S}$ under some distance function. The Voronoi diagram itself is the locus of all points $x \in \mathbb{R}^2$ such that x is closest to at least two sites in $\mathcal{S}$. The Voronoi diagram of a set of points under the Euclidean distance is the most well-known Voronoi diagram and has been studied in great detail since it was introduced by Shamos and Hoey [45]. Figure 4.1 shows an example of the Euclidean Voronoi diagram of a set of points. Voronoi diagrams of other geometric objects and distance functions have received a considerable amount of attention over the years. We refer the reader to the survey by Aurenhammer and Klein [5] and the book by Aurenhammer et al. [6] for more details.

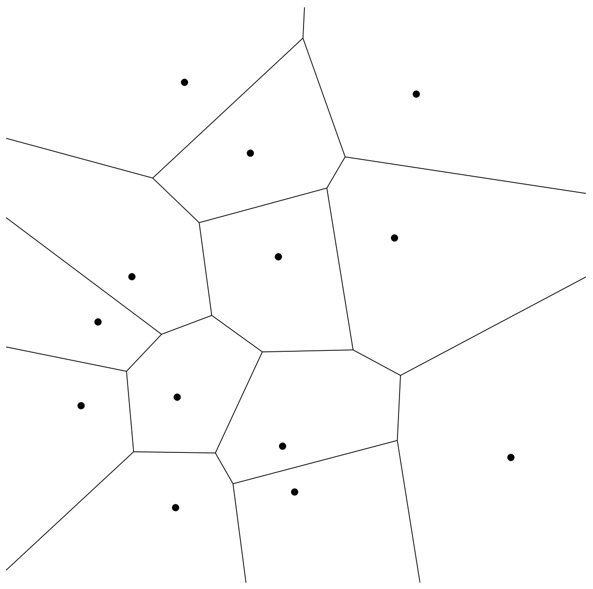


Figure 4.1: The Voronoi diagram of a set of points.

Let $\mathcal{P}$ be a simple polygon with n vertices. The Voronoi diagram of $\mathcal{P}$ is a Voronoi diagram whose sites are the open edges and reflex vertices of $\mathcal{P}$ [34]. Figure 4.2 shows an example of the Voronoi diagram of a simple polygon. We are only interested in the part of this diagram that is contained in $\mathcal{P}$, so we do not consider the parts of $\mathcal{V}(\mathcal{P})$ that lie outside of the polygon, and throughout the rest of this chapter, when we refer to $\mathcal{V}(\mathcal{P})$, we are referring to the parts of $\mathcal{V}(\mathcal{P})$ that are inside $\mathcal{P}$. The Voronoi diagram of $\mathcal{P}$ has size $O(n)$ and can be computed in $O(n)$ time [16].

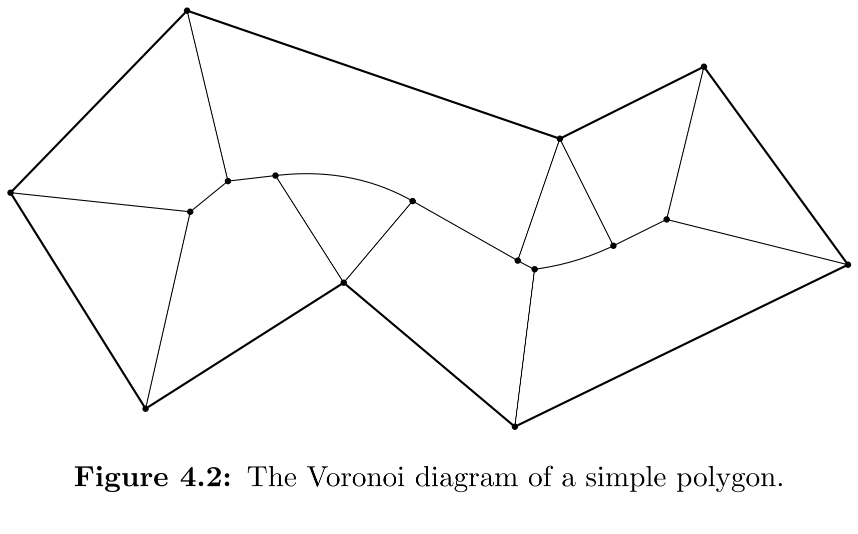


Figure 4.2: The Voronoi diagram of a simple polygon.

Combinatorially, $\mathcal{V}(\mathcal{P})$ is a tree-like planar graph with $O(n)$ edges. Each edge in $\mathcal{V}(\mathcal{P})$ is a line segment, or an arc of a parabola, that is a subset of the bisector of two Voronoi sites. The only points of $\partial\mathcal{P}$ in $\mathcal{V}(\mathcal{P})$ are the vertices of $\mathcal{P}$. Let x be a vertex of $\mathcal{P}$. If x is a non-reflex vertex then its degree in $\mathcal{V}(\mathcal{P})$ is one. Otherwise, x is a reflex vertex and its degree is two. If we treat each reflex vertex x as two distinct vertices, each of which is incident to one of the two edges of $\mathcal{V}(\mathcal{P})$ incident to x, then we may consider $\mathcal{V}(\mathcal{P})$ to be a tree whose leaves are the vertices of $\mathcal{P}$.

Let s and t be two vertices of $\mathcal{P}$. If s and t are non-reflex vertices then there is a unique path between s and t in $\mathcal{V}(\mathcal{P})$, otherwise there can be up to four paths between s and t. Let $e_1, \ldots, e_k$ be the edges of a path between s and t in $\mathcal{V}(\mathcal{P})$, chosen arbitrarily. Each edge e_i is an arc of the bisector of two Voronoi sites $\mathbf{p}_R \subseteq \mathcal{P}_{st}^R$ and $\mathbf{p}_L \subseteq \mathcal{P}_{st}^L$, each of which is a reflex vertex of $\mathcal{P}$ or an edge of $\mathcal{P}$. The bisector of $\mathbf{p}_R$ and $\mathbf{p}_L$ can consist of up to seven arcs of lines and parabolas, but e_i itself is either a line segment or an arc of a single parabola [34]. We define the *image* of e_i on $\mathbf{p}_R$, denoted by $I_R(e_i)$, as

$$I_R(e_i) = \bigcup_{x \in e_i} \{y \in \mathbf{p}_R : d(x,y) = \min_{z \in \overline{\mathbf{p}_R}} d(x,z)\},$$

where $\overline{\mathbf{p}_R}$ denotes the closure of $\mathbf{p}_R$ in the Euclidean topology. The image of e_i on $\mathbf{p}_L$, denoted by $I_L(e_i)$, is defined analogously. Figure 4.3 shows the four types of edges that can occur in $\mathcal{V}(\mathcal{P})$ and their images on $\partial\mathcal{P}$. In this figure, e_1 is incident to a reflex vertex and its images are equal to that vertex; the Voronoi sites defining e_2 are both reflex vertices; one site defining e_3 is a reflex vertex and the other is a line segment; and both sites defining e_4 are line segments. Figure 4.4 shows a Voronoi path between two vertices s and t in a simple polygon $\mathcal{P}$ (refer to Figure 4.2 for its Voronoi diagram). In Figure 4.4, the images of e_1, e_2 and e_3 on $\mathcal{P}_{st}^L$ are subsegments of a single edge of $\mathcal{P}$, as are the images of e_3, e_4 and e_5 on $\mathcal{P}_{st}^R$. Since t is a reflex vertex, $I_R(e_6) = I_L(e_6) = t$.

Lemma 4.1. *Let x be a point on an edge e_i of $\mathcal{V}(\mathcal{P})$ and let $\mathbf{d} = d(x, I_R(e_i)) = d(x, I_L(e_i))$. Then $\mathcal{D}(x, \mathbf{d}) \subseteq \mathcal{P}$.*

Proof. By definition, $d(x, \partial\mathcal{P}) = \mathbf{d}$, so $\mathcal{D}(x, \mathbf{d})$ cannot contain any part of $\partial\mathcal{P}$ in its interior. Therefore, $\mathcal{D}(x, \mathbf{d}) \subseteq \mathcal{P}$. □

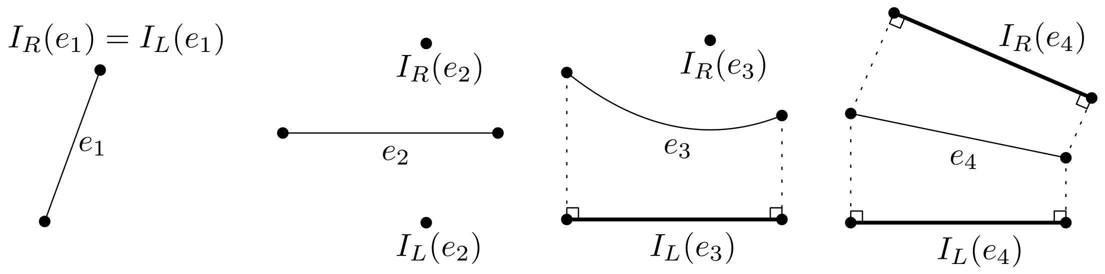


Figure 4.3: The images of four Voronoi edges e_1, e_2, e_3 and e_4.

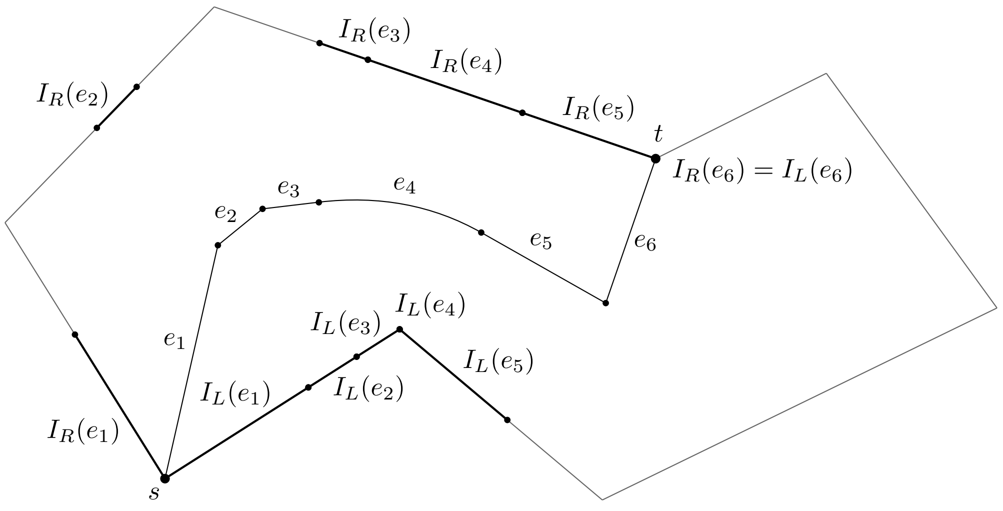


Figure 4.4: A path between s and t in $\mathcal{V}(\mathcal{P})$.

4.2 Hitting distance and path existence

Let x be a point in $\mathbb{R}^2$ and let $\mathbf{S}$ be a finite set of compact subsets of $\mathbb{R}^2$. We define the *hitting distance* between x and $\mathbf{S}$, denoted by $d_h(x, \mathbf{S})$, to be radius of the smallest disk $\mathcal{D}_h(x, \mathbf{S})$ centered at x that intersects every element of $\mathbf{S}$. The hitting distance is defined in terms of the Euclidean distance as

$$d_h(x, \mathbf{S}) = \max_{Y \in \mathbf{S}} \min_{y \in Y} \; d(x, y).$$

Let $\mathbf{S}_1$ and $\mathbf{S}_2$ be finite sets of compact subsets of $\mathbb{R}^2$. Then,

(i) $d_h(x, \mathbf{S}_1) < d_h(x, \mathbf{S}_2)$ if and only if $\mathcal{D}_h(x, \mathbf{S}_1) \subseteq \operatorname{int} \mathcal{D}_h(x, \mathbf{S}_2)$; and

(ii) $d_h(x, \mathbf{S}_1) = d_h(x, \mathbf{S}_2)$ if and only if $\mathcal{D}_h(x, \mathbf{S}_1) = \mathcal{D}_h(x, \mathbf{S}_2)$.

Let s and t be vertices of $\mathcal{P}$ and let $e_1, \ldots, e_k$ be the edges of the path between s and t in $\mathcal{V}(\mathcal{P})$. We define the *site* corresponding to e_i as $\mathbf{s}_i = \{I_R(e_i), I_L(e_i)\}$. For convenience, if an element of $\mathbf{s}_i$ consists of a single point then we abuse notation slightly and do not distinguish between the set containing that point and the point itself. There are four types of sites that can occur, each corresponding to one of the edges shown in Figure 4.3.

- e_i is an edge incident to a reflex vertex in $\mathcal{P}$. Then $\mathbf{s}_i$ contains a single element $I_R(e_i) = I_L(e_i)$ that is the reflex vertex incident to e_i, and we call $\mathbf{s}_i$ a *single-vertex site*. See edge e_1 in Figure 4.3.

- $I_R(e_i)$ and $I_L(e_i)$ are both reflex vertices of $\mathcal{P}$ but $I_R(e_i) \neq I_L(e_i)$. We call $\mathbf{s}_i$ a *vertex-vertex site*. See edge e_2 in Figure 4.3.

- $I_R(e_i)$ is a reflex vertex and $I_L(e_i)$ is a line segment, or $I_R(e_i)$ is a line segment and $I_L(e_i)$ is a reflex vertex. We call $\mathbf{s}_i$ a *vertex-segment site*. See edge e_3 in Figure 4.3.

- $I_R(e_i)$ and $I_L(e_i)$ are both line segments. We call $\mathbf{s}_i$ a *segment-segment site*. See edge e_4 in Figure 4.3.

Lemma 4.2. *Let e_i be an edge of the path between s and t in $\mathcal{V}(\mathcal{P})$ that is not incident to a reflex vertex of $\mathcal{P}$ and let $p_R \in I_R(e_i)$ and $p_L \in I_L(e_i)$. $\overline{p_R p_L}$ is a chord in $\mathcal{P}$.*

Proof. $\overline{p_R p_L}$ is contained in $\bigcup_{x\in e_i} \mathcal{D}_h(x, \mathbf{s}_i)$, which is contained in $\mathcal{P}$ due to Lemma 4.1. If $\mathbf{s}_i$ is a vertex-vertex site or a segment-segment site then $\bigcup_{x\in e_i} \mathcal{D}_h(x, \mathbf{s}_i)$ is convex so p_R and p_L are the only points of $\overline{p_R p_L}$ on its boundary, and thus the only points of $\overline{p_R p_L}$ on $\partial\mathcal{P}$. Otherwise, $\mathbf{s}_i$ is a vertex-segment site and there is a point x in e_i such that $p_R, p_L \in \mathcal{D}_h(x, \mathbf{s}_i)$. Since $\mathcal{D}_h(x, \mathbf{s}_i)$ is convex, p_R and p_L are the only points of $\overline{p_R p_R}$ on $\partial\mathcal{D}_h(x, \mathbf{s}_i)$ and thus the only points of $\overline{p_R p_L}$ on $\partial\mathcal{P}$. □

Let $\mathcal{S} = \{\mathbf{s}_i\}_{1\le i\le k}$ be the set of sites corresponding to the edges of the path between s and t in $\mathcal{V}(\mathcal{P})$ and let $\mathcal{O} = \mathcal{D}(c, r)$ be an obstacle. The hitting distances between c and the sites in $\mathcal{S}$ can be used to determine whether there is a obstacle-avoiding path between s and t.

Lemma 4.3. *There exists an obstacle-avoiding path between s and t if and only if $d_h(c, \mathbf{s}_i) > r$ for all $\mathbf{s}_i \in \mathcal{S}$.*

Proof. We start by proving the forwards direction. Suppose there is a site $\mathbf{s}_i$ in $\mathcal{S}$ such that $d_h(c, \mathbf{s}_i) \le r$. Then there exist points $p_R \in I_R(e_i)$ and $p_L \in I_L(e_i)$ such that $p_R, p_L \in \mathcal{O}$. By Lemma 4.2, $\overline{p_R p_L}$ is a chord in $\mathcal{P}$. Since $p_R \in \mathcal{P}^R_{st}$ and $p_L \in \mathcal{P}^L_{st}$, this chord separates s from t, and by the convexity of $\mathcal{O}$, $\overline{p_L p_R} \subseteq \mathcal{O}$. Therefore, there cannot be an obstacle-avoiding path between s and t.

We now prove the backwards direction. Let $\gamma_{st} = \bigcup_{1\le i\le k} e_i$ be the path between s and t in $\mathcal{V}(\mathcal{P})$. If $\gamma_{st} \cap \mathcal{O} = \emptyset$, then γ_{st} is an obstacle-avoiding path between s and t. If $\gamma_{st} \cap \mathcal{O}$ consists of a single connected component γ_{xy} with endpoints $x \prec_{\gamma_{st}} y$ then γ_{st} partitions $\mathcal{O}$ into two regions, $\mathcal{O}_R$, bounded by γ_{xy} and the clockwise arc α_R of $\partial\mathcal{O}$ between x and y; and $\mathcal{O}_L$, bounded by γ_{xy} and α_L, the counter-clockwise arc of $\partial\mathcal{O}$ between x and y. In order to prove that there is an obstacle-avoiding path between s and t it is sufficient to show that either α_R or α_L is contained in the interior of $\mathcal{P}$.

We claim that if $c \in \mathcal{O}_R$ then $\alpha_L \subseteq \operatorname{int}\mathcal{P}$, and if $c \in \mathcal{O}_L$ then $\alpha_R \subseteq \operatorname{int}\mathcal{P}$. Assume that $c \in \mathcal{O}_R$. The case where $c \in \mathcal{O}_L$ is symmetric. If α_L is not contained in the interior of $\mathcal{P}$, then α_L must contain at least one point z on the boundary of $\mathcal{P}$. Since $c \in \mathcal{O}_R$ and $z \in \mathcal{O}_L$, $\overline{cz}$ intersects γ_{xy} at a point c'. Let e_i be an edge of γ_{st} containing c' and let $p_R \in I_R(e_i)$ and $p_L \in I_L(e_i)$ be the points such that $d_h(c', \mathbf{s}_i) = d(c', p_R) = d(c', p_L)$. Refer to Figure 4.5. Then $d(c', p_R) \le d(c', z)$ and by the triangle inequality

$$\begin{aligned} d(c, p_R) &\le d(c, c') + d(c', p_R) \\ &\le d(c, c') + d(c', z) \\ &= d(c, z) \\ &= r. \end{aligned}$$

A symmetric argument shows that $d(c, p_L) \leq r$. Therefore, $p_R \in \mathcal{O}$ and $p_L \in \mathcal{O}$, implying that $d_h(c, \mathbf{s}_i) \leq r$. This contradicts the assumption that $d_h(c, \mathbf{s}_i) > r$ for all sites $\mathbf{s}_i \in \mathcal{S}$. Therefore, α_L is contained in the interior of $\mathcal{P}$ and there is an obstacle-avoiding path between s and t.

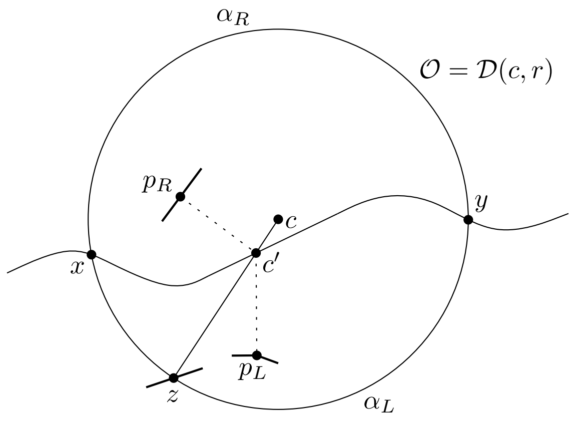


Figure 4.5: If $\alpha_L \cap \partial\mathcal{P} \neq \emptyset$ then $d(c, p_L) \leq r$ and $d(c, p_R) \leq r$.

In order to handle cases where $\gamma_{st} \cap \mathcal{O}$ consists of multiple connected components we show that each component can be considered separately. Let $\gamma_{x_1 y_1}, \ldots, \gamma_{x_\ell y_\ell}$ be the sequence of connected components of $\gamma_{st} \cap \mathcal{O}$, ordered by their minimum distance to s as measured along γ_{st}, where each $\gamma_{x_j y_j}$, $1 \leq j \leq \ell$, has endpoints $x_j \prec_{\gamma_{st}} y_j$. Let $s = z_1, z_2, \ldots, z_\ell, z_{\ell+1} = t$ be points on γ_{st} such that $y_j \prec_{\gamma_{st}} z_{j+1} \prec_{\gamma_{st}} x_{j+1}$ for all $1 \leq j \leq \ell$. Let $\gamma_{z_j z_{j+1}}$ denote the subpath of γ_{st} between z_j and z_{j+1}. Then $\gamma_{z_j z_{j+1}} \cap \mathcal{O}$ consists of a single connected component $\gamma_{x_j y_j}$, and if $d_h(c, \mathbf{s}_i) > r$ for all $\mathbf{s}_i \in \mathcal{S}$, then for all $1 \leq j \leq \ell$ there is an obstacle-avoiding path δ_j between z_j and z_{j+1}, and $\bigcup_{1 \leq j \leq \ell} \delta_j$ is an obstacle-avoiding path between s and t. ☐

To determine whether there is an obstacle-avoiding path between s and t it is sufficient to check whether the hitting distance between a site $\mathbf{s}_i$ in $\mathcal{S}$ that is closest to the center c of the disk $\mathcal{O} = \mathcal{D}(c, r)$ under the hitting distance is greater than r.

Corollary 4.4. *Let $\mathbf{s}_i$ be a site in $\mathcal{S}$ that is closest to the center c of disk obstacle $\mathcal{O} = \mathcal{D}(c, r)$ under the hitting distance. There is an obstacle-avoiding path between s and t if and only if $d_h(c, \mathbf{s}_i) > r$.*

Proof. This result follows immediately from Lemma 4.3. If $d_h(c, \mathbf{s}_i) \leq r$ then there is no obstacle-avoiding path between s and t. Otherwise, $d_h(c, \mathbf{s}_j) \geq d_h(c, \mathbf{s}_i) > r$ for all sites $\mathbf{s}_j \neq \mathbf{s}_i$ in $\mathcal{S}$, and there is an obstacle-avoiding path between s and t. ☐

4.3 The hitting distance Voronoi diagram

In order to find a site in $\mathcal{S}$ that is closest to the center c of a disk obstacle $\mathcal{O} = \mathcal{D}(c, r)$ under the hitting distance we define a Voronoi that we refer to as the *hitting distance Voronoi diagram* of $\mathcal{S}$ and denote by $\mathcal{V}_h(\mathcal{S})$. In this section, we examine the combinatorial structure of $\mathcal{V}_h(\mathcal{S})$ and show that it has size $O(k)$ and can be computed in $O(k \log k)$ time.

Voronoi diagrams of geometric sites under a variety of distance functions have received a considerable amount of attention over the years. Of particular interest is the *Hausdorff Voronoi diagram* of a finite set of compact sites, where the distance between a point $x \in \mathbb{R}^2$ and a site $\mathbf{S}$ is measured using the Hausdorff distance $d_H(x, \mathbf{S})$, defined as the maximum distance between x and an element of $\mathbf{S}$ [43, 18][1]. If each site $\mathbf{S}$ is a set of compact subsets of $\mathbb{R}^2$ then the Hausdorff distance can be defined as

$$d_H(x, \mathbf{S}) = \max_{Y \in \mathbf{S}} \max_{y \in Y} \; d(x, y),$$

which is similar to the hitting distance function defined in Section 4.2. If the sites are clusters of points then the Hausdorff Voronoi diagram has been referred to as the *Voronoi diagram of point clusters* [19], or the *Min-Max Voronoi diagram* [41, 42]. If the sites are disjoint convex sets then the diagram is known as the *closest covered set diagram* [1]. In general, the Hausdorff distance Voronoi diagram of m sites has complexity $O(m^2\alpha(m))$, where $\alpha(m)$ is the inverse Ackermann function [19]. We say that two sites $\mathbf{s}_i$ and $\mathbf{s}_j$ are *non-crossing* if $\mathcal{CH}(\mathbf{s}_i) \setminus \mathcal{CH}(\mathbf{s}_j)$ consists of at most one connected component, where $\mathcal{CH}(\mathbf{S})$ denotes the convex hull of the elements of a set $\mathbf{S}$. A set of sites is non-crossing if its sites are pairwise non-crossing. The complexity of the Hausdorff Voronoi diagram of a set of non-crossing sites is only $O(m)$ [19]. Since the hitting distance is similar to the Hausdorff distance, and the sites in $\mathcal{S}$ are pairwise non-crossing due to Lemma 4.2, we will use an approach similar to that of Abellanas et al. [1] to show that $\mathcal{V}_h(\mathcal{S})$ has linear size.

Specifically, we show that $\mathcal{V}_h(\mathcal{S})$ fits into the well-known *abstract Voronoi diagram* framework introduced by Klein [30] and later refined by Klein et al. [31]. The abstract

[1]The Hausdorff distance between two sets X and Y is defined as

$$d_H(X, Y) = \max\{\sup_{x \in X} \inf_{y \in Y} \; d(x, y), \sup_{y \in Y} \inf_{x \in X} \; d(x, y)\}.$$

In this case however, $X = \{x\}$, and Y, as well as its elements, are compact sets, which is why we are able to simplify the definition of the Hausdorff distance to the maximum distance between x and an element of Y.

Voronoi diagram framework generalizes the idea of a Voronoi diagrams by defining regions in terms of systems of bisecting curves instead of in terms of a distance function. If the Voronoi diagram of a set of m sites is an abstract Voronoi diagram then its complexity is linear in m, and if the bisecting curves that define the diagram have constant complexity, then the diagram can be constructed in $O(m \log m)$ time [30]. Many Voronoi diagrams that are defined in terms of distance functions are also abstract Voronoi diagrams. Notably, every Voronoi diagram of a point set where distance is measured using a strictly convex distance function is an abstract Voronoi diagram [36]. Examples of abstract Voronoi diagrams that explicitly make use of Klein's framework include the previously mentioned closest covered set diagram of Abellanas [1], and the transportation network Voronoi diagrams introduced by Aicholzer et al. [2] and further studied by Bae and Chwa [7].

We now define the hitting distance Voronoi diagram of $\mathcal{S}$. Let $\mathbf{s}_i$ and $\mathbf{s}_j$ be sites in $\mathcal{S}$. Following the notation of Icking et al. [28] and Aichholzer et al. [2], the *hitting distance dominance region* of $\mathbf{s}_i$ with respect to $\mathbf{s}_j$ is defined as

$$D_h(\mathbf{s}_i, \mathbf{s}_j) = \{x \in \mathbb{R}^2 : d_h(x, \mathbf{s}_i) < d_h(x, \mathbf{s}_j)\},$$

and the *hitting distance bisector* of $\mathbf{s}_i$ and $\mathbf{s}_j$ is defined as

$$B_h(\mathbf{s}_i, \mathbf{s}_j) = \{x \in \mathbb{R}^2 : d_h(x, \mathbf{s}_i) = d_h(x, \mathbf{s}_j)\}.$$

The hitting distance bisector of $\mathbf{s}_i$ and $\mathbf{s}_j$ is composed of pieces of the Euclidean bisectors between the pairs of elements from $\mathbf{s}_i$ and $\mathbf{s}_j$. Let $B(x, y)$ denote the Euclidean bisector of x and y. Let $H_R(e_i)$ be the set of points $x \in \mathbb{R}^2$ such that $d(x, I_R(e_i)) \geq d(x, I_L(e_i))$. We define $H_L(e_i)$ symmetrically as the set of points $x \in \mathbb{R}^2$ such that $d(x, I_L(e_i)) \geq d(x, I_R(e_i))$. The parts of $B_h(\mathbf{s}_i, \mathbf{s}_j)$ contained in the regions $H_R(e_i) \cap H_R(e_j)$, $H_R(e_i) \cap H_L(e_j)$, $H_L(e_i) \cap H_R(e_j)$ and $H_L(e_i) \cap H_L(e_j)$ are the subsets of $B(I_R(e_i), I_R(e_j))$, $B(I_R(e_i), I_L(e_i))$, $B(I_L(e_i), I_R(e_i))$, and $B(I_L(e_i), I_L(e_i))$ contained in these regions, respectively. Figure 4.6 shows the hitting distance bisector $B_h(\mathbf{s}_2, \mathbf{s}_4)$ of the two sites $\mathbf{s}_2$ and $\mathbf{s}_4$ associated with the edges e_2 and e_4 of the Voronoi path shown in Figure 4.4. This bisector consists of pieces of the Euclidean bisectors shown in Figure 4.7.

The hitting distance bisector of two sites can contain a 2-dimensional region if an element of $\mathbf{s}_i$ intersects an element $\mathbf{s}_j$. For instance, the part of $B_h(\mathbf{s}_i, \mathbf{s}_j)$ belonging to $B(I_R(e_i), I_R(e_j))$ can contain a 2-dimensional region if $I_R(e_i) \cap I_R(e_j) \neq \emptyset$. There are two cases where the elements of $\mathbf{s}_i$ and $\mathbf{s}_j$ can intersect.

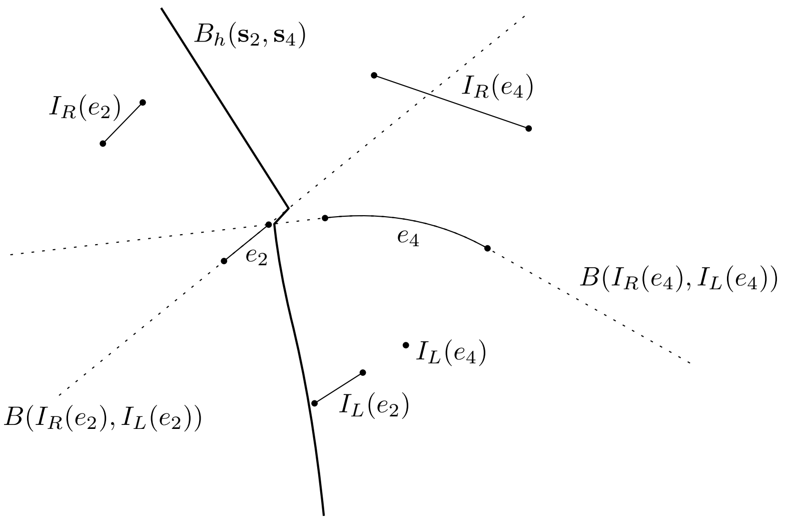


Figure 4.6: The hitting distance bisector $B_h(\mathbf{s}_2, \mathbf{s}_4)$ of the two sites $\mathbf{s}_2$ and $\mathbf{s}_4$ associated with the edges e_2 and e_4 of the Voronoi path shown in Figure 4.4.

- $\mathbf{s}_i$ and $\mathbf{s}_j$ share a point p on the interior of an edge $\overline{uv}$ of $\mathcal{P}$. Without loss of generality, assume that $p \in I_R(e_i) \cap I_R(e_j)$. Then $I_R(e_i)$ and $I_R(e_j)$ are subsegments of an edge of $\mathcal{O}$ and p is an endpoint of both segments in the interior of this edge. In this case, the region $\{x \in \mathbb{R}^2 : d_h(x, \mathbf{s}_i) = d_h(x, \mathbf{s}_j) = d(x, p)\}$ is a subset of the line perpendicular to $\overline{uv}$ containing p, so $B_h(\mathbf{s}_i, \mathbf{s}_j)$ does not contain a 2-dimensional region. Refer to Figure 4.8.

- $\mathbf{s}_i$ and $\mathbf{s}_j$ share a point p that is a reflex vertex of $\mathcal{P}$. Without loss of generality, assume that $p \in I_R(e_i) \cap I_R(e_j)$. In this case, the region $\{x \in \mathbb{R}^2 : d_h(x, \mathbf{s}_i) = d_h(x, \mathbf{s}_j) = d(x, p)\}$ is the 2-dimensional cone with apex p bounded by the two lines through p that are perpendicular to $I_R(e_i)$ and $I_R(e_j)$ and opening into the interior of $\mathcal{P}$. Refer to Figure 4.9.

  A special case occurs when both elements of $\mathbf{s}_i$ are reflex vertices p_R and p_L such that $p_R \in I_R(e_j)$ and $p_L \in I_L(e_j)$. In this case, the hitting distance bisector consists solely of the 2-dimensional region $\{x \in \mathbb{R}^2 : d_h(x, \mathbf{s}_i) = d_h(x, \mathbf{s}_j) = d(x, p_R)\} \cup \{x \in \mathbb{R}^2 : d_h(x, \mathbf{s}_i) = d_h(x, \mathbf{s}_j) = d(x, p_L)\}$. See Figure 4.10 for an example. In this case, $D_h(\mathbf{s}_i, \mathbf{s}_j) = \emptyset$.

Bisectors with 2-dimensional regions are difficult to deal with when constructing Voronoi diagrams. Several other distance functions that give bisectors with 2-dimensional regions include the geodesic distance function [4], non-strict convex

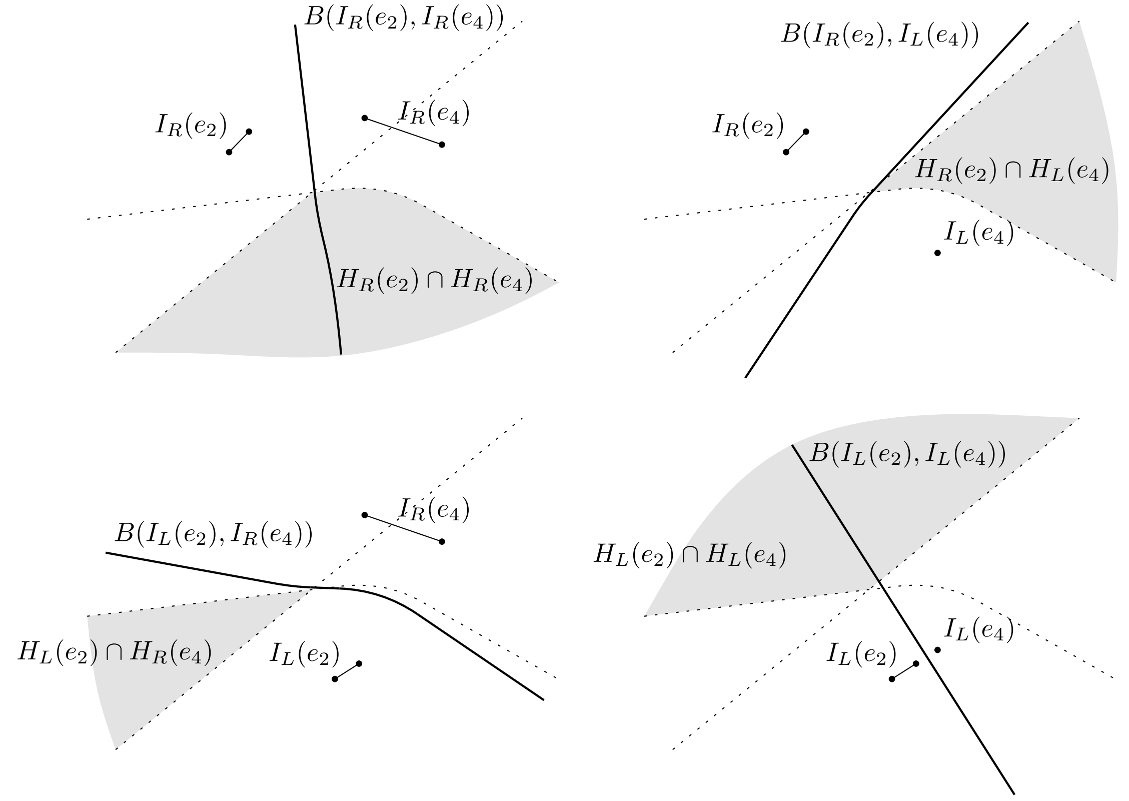


Figure 4.7: The bisectors that make up $B_h(\mathbf{s}_2, \mathbf{s}_4)$. The shaded regions indicate which pieces of these bisectors end up forming $B_h(\mathbf{s}_2, \mathbf{s}_4)$, shown in Figure 4.6.

distance functions [28] and the transportation network distance function [2, 7]. A common approach to dealing with 2-dimensional bisectors is to define an ordering $\prec$ on the elements of $\mathcal{S}$. If $\mathbf{s}_i \prec \mathbf{s}_j$ then we assign the 2-dimensional region to the Voronoi region of $\mathbf{s}_i$, otherwise we assign it to the Voronoi region of $\mathbf{s}_j$. We then define a bisecting curve $B^\star_h(\mathbf{s}_i, \mathbf{s}_j)$, referred to as the *chosen hitting distance bisector* of $\mathbf{s}_i$ and $\mathbf{s}_j$, that follows the boundary of this new Voronoi region. The chosen hitting distance bisector of $\mathbf{s}_i$ and $\mathbf{s}_j$ does not contain a 2-dimensional region, so it can be used in place of the actual bisector of $\mathbf{s}_i$ and $\mathbf{s}_j$.

We have to be careful with how we define the ordering on the sites in $\mathcal{S}$. If $B_h(\mathbf{s}_i, \mathbf{s}_j)$ contains a 2-dimensional region, and both elements of $\mathbf{s}_i$ are distinct reflex vertices that are also contained in the elements of $\mathbf{s}_j$, then defining $\prec$ such that $\mathbf{s}_j \prec \mathbf{s}_i$ results in a degenerate chosen hitting distance bisector $B^\star_h(\mathbf{s}_i, \mathbf{s}_j) = \emptyset$. Informally, we want vertex-vertex sites to appear at the beginning of the ordering, so that each Voronoi region is non-empty, and we want single-vertex sites to appear at the end of the

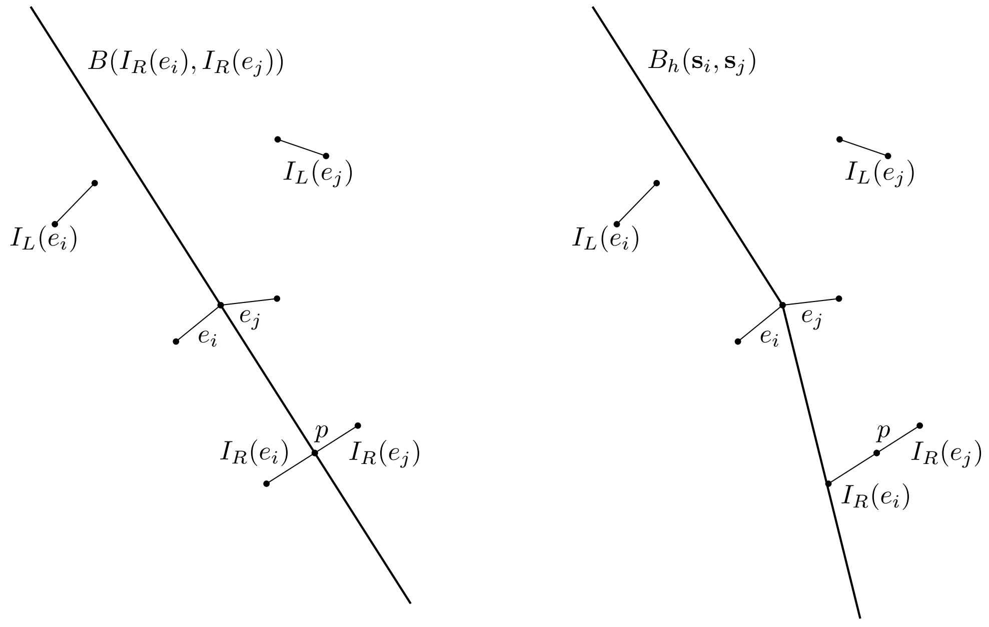


Figure 4.8: The Euclidean bisector of $I_R(e_i)$ and $I_R(e_j)$ and the hitting distance bisector of $\mathbf{s}_i$ and $\mathbf{s}_j$.

ordering, so that each Voronoi edge is contained in its own Voronoi region. This is satisfied by defining $\prec$ so that $\mathbf{s}_i \prec \mathbf{s}_j$ if and only if

(i) $\mathbf{s}_i$ is a vertex-vertex site and $\mathbf{s}_j$ is not a vertex-vertex site; or

(ii) $\mathbf{s}_i$ is a not a single-vertex site and $\mathbf{s}_j$ is a single-vertex site; or

(iii) $i \leq j$ and we are not in case (i) or (ii).

Figure 4.11 shows the possible chosen hitting distance bisectors of the sites $\mathbf{s}_i$ and $\mathbf{s}_j$, depicted in Figure 4.9, if $\mathbf{s}_i \prec \mathbf{s}_j$ or if $\mathbf{s}_j \prec \mathbf{s}_i$.

The *hitting distance Voronoi region* of $\mathbf{s}_i$ with respect to $\mathbf{s}_j$ is defined as

$$R_h(\mathbf{s}_i, \mathbf{s}_j) = \begin{cases} D_h(\mathbf{s}_i, \mathbf{s}_j) \cup B_h(\mathbf{s}_i, \mathbf{s}_j) & \text{if } \mathbf{s}_i \prec \mathbf{s}_j, \\ D_h(\mathbf{s}_i, \mathbf{s}_j) & \text{if } \mathbf{s}_j \prec \mathbf{s}_i, \end{cases}$$

and with respect to a set of sites $\mathcal{S}' \subseteq \mathcal{S}$ as

$$R_h(\mathbf{s}_i, \mathcal{S}') = \bigcap_{\mathbf{s}_j \in \mathcal{S}', \mathbf{s}_j \neq \mathbf{s}_i} R_h(\mathbf{s}_i, \mathbf{s}_j).$$

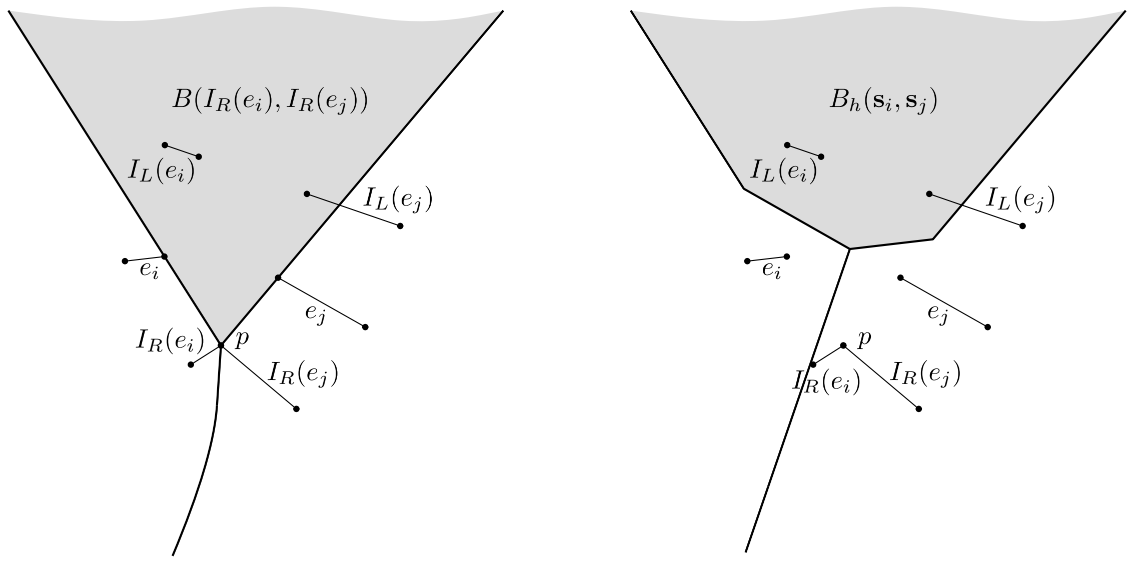


Figure 4.9: The Euclidean bisector of $I_R(e_i)$ and $I_R(e_j)$ and the hitting distance bisector of $\mathbf{s}_i$ and $\mathbf{s}_j$. Both bisectors contain 2-dimensional regions, shown in grey.

The chosen hitting distance bisector of $\mathbf{s}_i$ and $\mathbf{s}_j$ is defined as

$$B_h^\star(\mathbf{s}_i, \mathbf{s}_j) = \partial R_h(\mathbf{s}_i, \mathbf{s}_j),$$

which is equivalent to $B_h(\mathbf{s}_i, \mathbf{s}_j)$ if the elements of $\mathbf{s}_i$ and $\mathbf{s}_j$ do not intersect. Finally, the *hitting distance Voronoi diagram* of $\mathcal{S}$ is defined as

$$\mathcal{V}_h(\mathcal{S}) = \bigcup_{\mathbf{s}_i \in \mathcal{S}} \partial R_h(\mathbf{s}_i, \mathcal{S}).$$

In order to prove that the hitting distance Voronoi diagram is an abstract Voronoi diagram, as defined by Klein et al. [31], we must show that the following properties are satisfied.

(i) For any pair of sites $\mathbf{s}_i, \mathbf{s}_j \in \mathcal{S}$ such that $\mathbf{s}_i \neq \mathbf{s}_j$, $B_h^\star(\mathbf{s}_i, \mathbf{s}_j)$ is homeomorphic to a line.

(ii) For all non-empty subsets $\mathcal{S}' \subseteq \mathcal{S}$ and for all sites $\mathbf{s}_i \in \mathcal{S}'$, $R_h(\mathbf{s}_i, \mathcal{S}')$ is path-connected.

(iii) For all non-empty subsets $\mathcal{S}' \subseteq \mathcal{S}$,

$$\bigcup_{\mathbf{s}_i \in \mathcal{S}'} R_h(\mathbf{s}_i, \mathcal{S}') = \mathbb{R}^2.$$

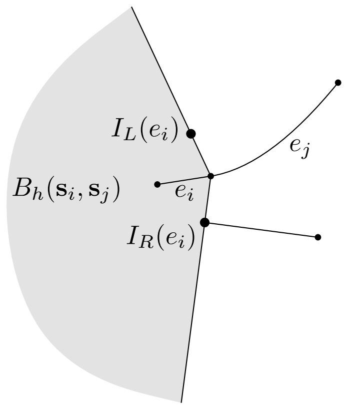


Figure 4.10: A hitting distance bisector that consists entirely of a 2-dimensional region.

(iv) For any four sites $\mathbf{s}_i, \mathbf{s}_j, \mathbf{s}_{i'}$ and $\mathbf{s}_{j'}$ in $\mathcal{S}$, where $\mathbf{s}_i \neq \mathbf{s}_j$, $\mathbf{s}_{i'} \neq \mathbf{s}_{j'}$, $B_h^\star(\mathbf{s}_i, \mathbf{s}_j) \cap B_h^\star(\mathbf{s}_{i'}, \mathbf{s}_{j'})$ consists of a finite number of connected components.

The remainder of this section is devoted to showing that these four properties are satisfied and, thus, that $\mathcal{V}_h(\mathcal{S})$ is abstract Voronoi diagram.

Lemma 4.5. *Let $\mathbf{s}_i$ be a site in $\mathcal{S}$ and let x be a point in $\mathbb{R}^2$. There exists a path between x and e_i such that for every point y on this path and for every site $\mathbf{s}_j \in \mathcal{S}$ such that $\mathbf{s}_j \neq \mathbf{s}_i$,*

(i) *if $d_h(x, \mathbf{s}_i) < d_h(x, \mathbf{s}_j)$, then $d_h(y, \mathbf{s}_i) < d_h(y, \mathbf{s}_j)$; and*

(ii) *if $d_h(x, \mathbf{s}_i) = d_h(x, \mathbf{s}_j)$, then $d_h(y, \mathbf{s}_i) \leq d_h(y, \mathbf{s}_j)$.*

Proof. Let p be a point in an element of $\mathbf{s}_i$ such that $d_h(x, \mathbf{s}_i) = d(x, p)$. Without loss of generality, assume that $p \in I_R(e_i)$. We define the *strip* of a line segment $\overline{uv}$, denoted by strip($\overline{uv}$), as the region, bounded by the lines perpendicular to $\overline{uv}$ passing through u and v, that contains $\overline{uv}$. We distinguish between four cases.

- $I_R(e_i)$ is a line segment and $x \in \text{strip}(I_R(e_i))$. Then $\overline{xp}$ crosses e_i at a point z. Refer to Figure 4.12. We claim that $\overline{xz}$ satisfies conditions (i) and (ii).

  (i) Let $\mathbf{s}_j \neq \mathbf{s}_i$ be a site in $\mathcal{S}$. If $d_h(x, \mathbf{s}_i) < d_h(x, \mathbf{s}_j)$ then $\mathcal{D}_h(x, \mathbf{s}_i)$ cannot intersect both elements of $\mathbf{s}_j$. Let y be a point in $\overline{xz}$. By Lemma 2.5, $\mathcal{D}_h(y, \mathbf{s}_i) \subseteq \mathcal{D}_h(x, \mathbf{s}_i)$, and $\mathcal{D}_h(y, \mathbf{s}_i)$ cannot intersect both elements of $\mathbf{s}_j$ either. Therefore, $d_h(y, \mathbf{s}_i) < d_h(y, \mathbf{s}_j)$.

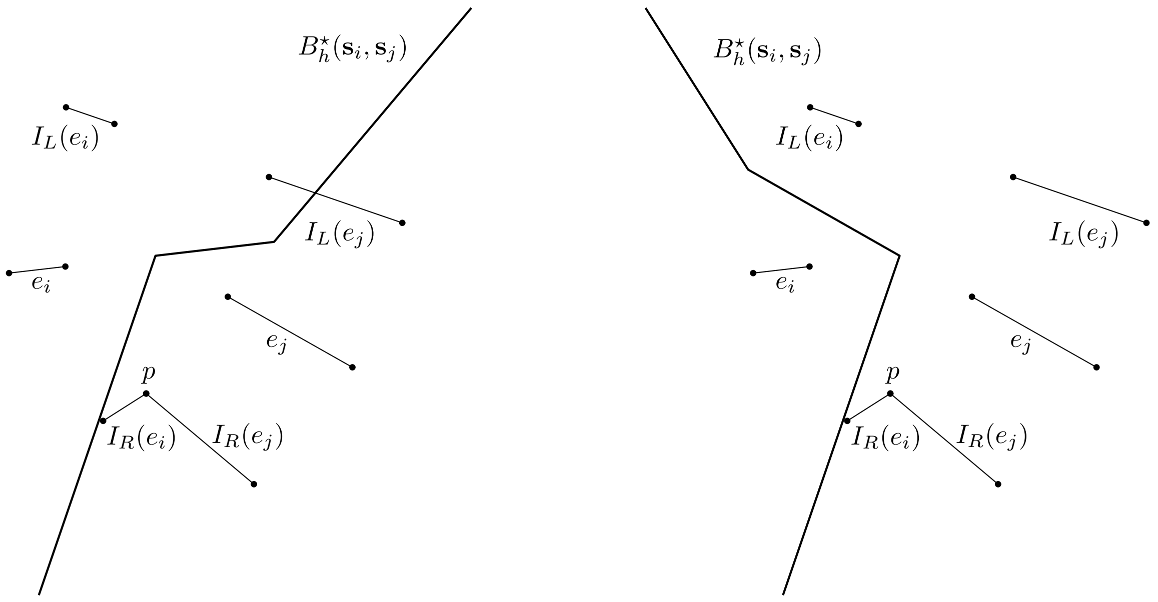


Figure 4.11: The chosen hitting distance bisector of sites $\mathbf{s}_i$ and $\mathbf{s}_j$ in Figure 4.9 if $\mathbf{s}_i \prec \mathbf{s}_j$ (left) or if $\mathbf{s}_j \prec \mathbf{s}_i$ (right).

(ii) Let $\mathbf{s}_j \neq \mathbf{s}_i$ be a site in $\mathcal{S}$ such that $d_h(x, \mathbf{s}_i) = d_h(x, \mathbf{s}_j)$. Assume that there is a point $y \in \overline{xz}$ such that $d_h(y, \mathbf{s}_i) > d_h(y, \mathbf{s}_j)$. Then $\operatorname{int} \mathcal{D}_h(y, \mathbf{s}_i)$ intersects both elements of $\mathbf{s}_j$ and, since $\mathcal{D}_h(y, \mathbf{s}_i) \subseteq \mathcal{D}_h(x, \mathbf{s}_i)$, $\operatorname{int} \mathcal{D}_h(x, \mathbf{s}_i)$ also intersects both elements of $\mathbf{s}_j$. This implies that $d_h(x, \mathbf{s}_i) > d_h(x, \mathbf{s}_j)$, which is a contradiction. Therefore, $d_h(y, \mathbf{s}_i) \leq d_h(y, \mathbf{s}_j)$.

- $I_R(e_i)$ is a line segment and $x \notin \operatorname{strip}(I_R(e_i))$. Without loss of generality, assume that $I_R(e_i)$ is a horizontal line segment and x lies to the right of $I_R(e_i)$. Let p be the rightmost endpoint of $I_R(e_i)$ and let z be the point of e_i such that $d_h(z, \mathbf{s}_i) = d(z, p)$. Then z is also the rightmost endpoint of e_i. We claim that $\overline{xz}$ satisfies conditions (i) and (ii).

(i) Let $\mathbf{s}_j \neq \mathbf{s}_i$ be a site in $\mathcal{S}$ such that $d_h(x, \mathbf{s}_i) < d_h(x, \mathbf{s}_j)$ and let y be a point in $\overline{xz}$. The region of the plane where the hitting distance to $\mathbf{s}_i$ is equal to the Euclidean distance to $I_R(e_i)$ is a convex set, so $d_h(y, \mathbf{s}_i) = d(y, I_R(e_i)) = d(y, p)$. Since x, y and z are collinear, by Lemma 2.7 we have that $\mathcal{D}_h(y, \mathbf{s}_i) \subseteq \mathcal{D}_h(x, \mathbf{s}_i) \cup \mathcal{D}_h(z, \mathbf{s}_i)$ and $\mathcal{D}_h(y, \mathbf{s}_i) \setminus \mathcal{D}_h(x, \mathbf{s}_i) \subseteq \operatorname{int} \mathcal{D}_h(z, \mathbf{s}_i)$. Refer to Figure 4.13.

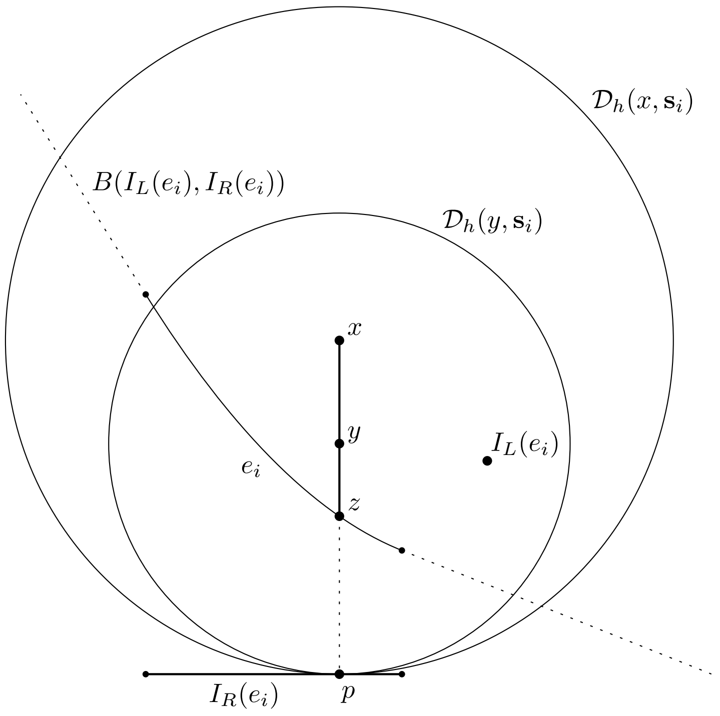

$\mathcal{D}_h(x, \mathbf{s}_i)$
$B(I_L(e_i), I_R(e_i))$
$\mathcal{D}_h(y, \mathbf{s}_i)$
x
y
$I_L(e_i)$
e_i
z
$I_R(e_i)$
p

Figure 4.12: A path satisfying conditions (i) and (ii) of Lemma 4.5 when $I_R(e_i)$ is a line segment and $x \in \text{strip}(I_R(e_i))$.

Assume that $d_h(y, \mathbf{s}_i) \geq d_h(y, \mathbf{s}_j)$. Then $\mathcal{D}_h(y, \mathbf{s}_j) \subseteq \operatorname{int} \mathcal{D}_h(y, \mathbf{s}_i)$, and we have that $\mathcal{D}_h(y, \mathbf{s}_j) \subseteq \operatorname{int}(\mathcal{D}_h(x, \mathbf{s}_i) \cup \mathcal{D}_h(z, \mathbf{s}_i))$ and there exist points $p_R \in I_R(e_j)$ and $p_L \in I_L(e_j)$ such that $p_R, p_L \in \mathcal{D}_h(y, \mathbf{s}_j)$. At least one of p_R and p_L must be contained in $\mathcal{D}_h(y, \mathbf{s}_i) \backslash \mathcal{D}_h(x, \mathbf{s}_i)$, or else $d_h(x, \mathbf{s}_i) \geq d_h(x, \mathbf{s}_j)$, so $p_R \in \operatorname{int} \mathcal{D}_h(z, \mathbf{s}_i)$ or $p_L \operatorname{int} \mathcal{D}_h(z, \mathbf{s}_i)$. Either case leads to a contradiction, however, since $\mathcal{D}_h(z, \mathbf{s}_i)$ cannot contain any part of $\partial \mathcal{P}$ in its interior, by Lemma 4.1. Therefore, $d_h(y, \mathbf{s}_i) < d_h(y, \mathbf{s}_j)$.

(ii) Let $\mathbf{s}_j \neq \mathbf{s}_i$ be a site in $\mathcal{S}$ such that $d_h(x, \mathbf{s}_i) = d_h(x, \mathbf{s}_j)$ and let y be a point in $\overline{xz}$. By assuming that $d_h(y, \mathbf{s}_i) > d_h(y, \mathbf{s}_j)$ and applying the same argument used previously to show that condition (i) holds, we arrive at a contradiction, proving that $d_h(y, \mathbf{s}_i) \leq d_h(y, \mathbf{s}_j)$.

- $I_R(e_i)$ is a reflex vertex and $\overline{xp}$ intersects e_i. Let z be the point where $\overline{xp}$ intersects e_i. See Figure 4.14. Using an argument similar to the case where $I_R(e_i)$ is a line segment and $x \in \text{strip}(I_R(e_i))$, it can be shown that $\overline{xz}$ satisfies conditions (i) and (ii).

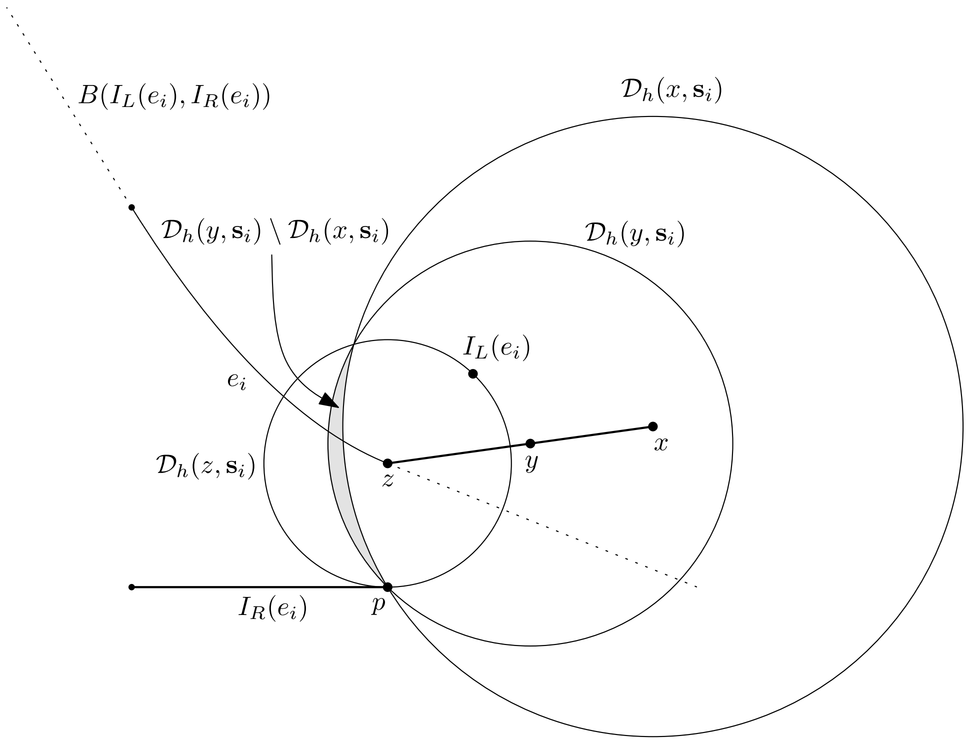

$B(I_L(e_i), I_R(e_i))$
$\mathcal{D}_h(x, \mathbf{s}_i)$
$\mathcal{D}_h(y, \mathbf{s}_i) \setminus \mathcal{D}_h(x, \mathbf{s}_i)$
$\mathcal{D}_h(y, \mathbf{s}_i)$
$I_L(e_i)$
e_i
x
$\mathcal{D}_h(z, \mathbf{s}_i)$
z
y
$I_R(e_i)$
p

Figure 4.13: A path satisfying conditions (i) and (ii) of Lemma 4.5 when $I_R(e_i)$ is a line segment and $x \notin \operatorname{strip}(I_R(e_i))$.

- $I_R(e_i)$ is a reflex vertex and $\overline{xp}$ does not intersect e_i. Then $\overline{xp}$ intersects $B(I_R(e_i), I_L(e_i))$ at a point x' and we take z to be the closest point of e_i to x' as measured by the distance along the bisector $B(I_R(e_i), I_L(e_i))$. See Figure 4.15. Using an argument similar to the one used in the case where $I_R(e_i)$ is a line segment and $x \notin \operatorname{strip}(I_R(e_i))$, it can be shown that $\overline{xz}$ satisfies conditions (i) and (ii). □

Lemma 4.6. $\operatorname{int} e_i \subseteq R_h(\mathbf{s}_i, \mathcal{S})$.

Proof. Let x be a point in $\operatorname{int} e_i$. Then $\mathcal{D}_h(x, \mathbf{s}_i)$ does not contain any part of $\partial\mathcal{P}$ in its interior and touches $\partial\mathcal{P}$ at at most two points, $p_R \in I_R(e_i)$ and $p_L \in I_L(e_i)$ [34]. Therefore, there cannot be a site $\mathbf{s}_j \in \mathcal{S}$ such that $d_h(x, \mathbf{s}_i) > d_h(x, \mathbf{s}_j)$. Let $\mathbf{s}_j \neq \mathbf{s}_i$ be another site in $\mathcal{S}$. If $d_h(x, \mathbf{s}_i) < d_h(x, \mathbf{s}_j)$ then $x \in D_h(\mathbf{s}_i, \mathbf{s}_j)$ and $x \in R_h(\mathbf{s}_i, \mathbf{s}_j)$. Otherwise, if $d_h(x, \mathbf{s}_i) = d_h(x, \mathbf{s}_j)$ then $x \in R_h(\mathbf{s}_i, \mathbf{s}_j)$ if and only if $\mathbf{s}_i \prec \mathbf{s}_j$. Since $R_h(\mathbf{s}_i, \mathcal{S})$ is defined as $\bigcap_{\mathbf{s}_j \in \mathcal{S}, \mathbf{s}_j \neq \mathbf{s}_i} R_h(\mathbf{s}_i, \mathbf{s}_j)$, we must show that if there is a site $\mathbf{s}_j$ such that $d_h(x, \mathbf{s}_i) = d_h(x, \mathbf{s}_j)$, then $\mathbf{s}_i \prec \mathbf{s}_j$. We distinguish between four cases based on the type of site that $\mathbf{s}_i$ can be.

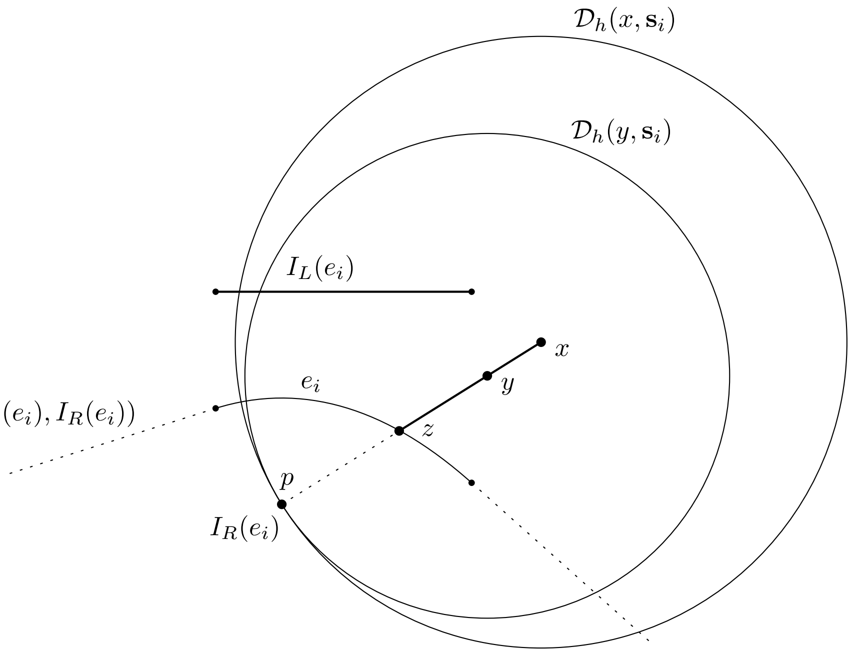

$\mathcal{D}_h(x, \mathbf{s}_i)$
$\mathcal{D}_h(y, \mathbf{s}_i)$
$I_L(e_i)$
x
y
e_i
$B(I_L(e_i), I_R(e_i))$
z
p
$I_R(e_i)$

Figure 4.14: A path satisfying conditions (i) and (ii) of Lemma 4.5 when $I_R(e_i)$ is a reflex vertex and $\overline{xp}$ intersects e_i.

- $\mathbf{s}_i$ is a single-vertex site. Then $\mathcal{D}_h(x, \mathbf{s}_i)$ touches $\partial\mathcal{P}$ at a single point and there cannot be a site $\mathbf{s}_j \neq \mathbf{s}_i$ in $\mathcal{S}$ such that $d_h(x, \mathbf{s}_i) = d_h(x, \mathbf{s}_j)$.

- $\mathbf{s}_i$ is a vertex-vertex site. Suppose that there is a site $\mathbf{s}_j \neq \mathbf{s}_i$ in $\mathcal{S}$ such that $d_h(x, \mathbf{s}_j) = d_h(x, \mathbf{s}_i)$. The elements of $\mathbf{s}_j$ cannot be distinct reflex vertices of $\mathcal{P}$ as well, or else $\mathbf{s}_j = \mathbf{s}_i$, so $\mathbf{s}_j$ is not a vertex-vertex site. Then $\mathbf{s}_i \prec \mathbf{s}_j$.

- $\mathbf{s}_i$ is vertex-segment site. Without loss of generality, assume that $I_R(e_i)$ is a line segment and $I_L(e_i)$ is a reflex vertex. If there is a site $\mathbf{s}_j$ such that $d_h(x, \mathbf{s}_i) = d_h(x, \mathbf{s}_j)$ then $\mathbf{s}_j$ is a single-vertex site such that $I_R(\mathbf{s}_j) = I_L(\mathbf{s}_j) = p_L$, since p_R is not an endpoint of the line segment $I_R(e_i)$ and thus cannot belong to an element of any site other than $\mathbf{s}_i$. Then $\mathbf{s}_i \prec \mathbf{s}_j$.

- $\mathbf{s}_i$ is a segment-segment site. Only the endpoints of $I_R(e_i)$ and $I_L(e_i)$ can intersect the elements of another site, so p_R and p_L cannot belong to the elements of another site. Therefore, there cannot be a site $\mathbf{s}_j \neq \mathbf{s}_i$ in $\mathcal{S}$ such that $d_h(x, \mathbf{s}_i) = d_h(x, \mathbf{s}_j)$. $\square$

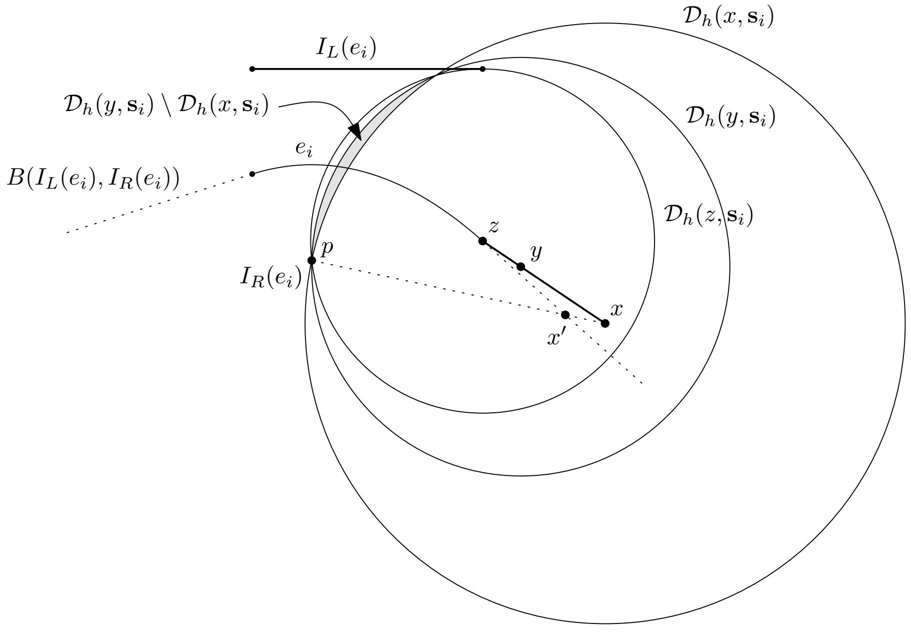

$\mathcal{D}_h(x, \mathbf{s}_i)$
$I_L(e_i)$
$\mathcal{D}_h(y, \mathbf{s}_i) \setminus \mathcal{D}_h(x, \mathbf{s}_i)$
$\mathcal{D}_h(y, \mathbf{s}_i)$
e_i
$B(I_L(e_i), I_R(e_i))$
$\mathcal{D}_h(z, \mathbf{s}_i)$
z
p
y
$I_R(e_i)$
x
x'

Figure 4.15: A path satisfying conditions (i) and (ii) of Lemma 4.5 when $I_R(e_i)$ is a reflex vertex and $\overline{xp}$ does not intersect e_i.

Lemma 4.7. *$R_h(\mathbf{s}_i, \mathcal{S})$ is path-connected.*

Proof. Let x and y be points in $R_h(\mathbf{s}_i, \mathcal{S})$. By definition, for all sites $\mathbf{s}_j \neq \mathbf{s}_i$ in $\mathcal{S}$, if $\mathbf{s}_j \prec \mathbf{s}_i$ then $d_h(x, \mathbf{s}_i) < d_h(x, \mathbf{s}_j)$, and if $\mathbf{s}_i \prec \mathbf{s}_j$ then $d_h(x, \mathbf{s}_i) \leq d_h(x, \mathbf{s}_j)$. Then by Lemma 4.5 there is a path $\gamma_{xx'}$ between x and a point $x' \in e_i$ such that $\gamma_{xx'} \subseteq R_h(\mathbf{s}_i, \mathcal{S})$. Similarly, there is a path $\gamma_{yy'} \subseteq R_h(\mathbf{s}_i, \mathcal{S})$ between y and a point $y' \in e_i$. By Lemma 4.6, the arc $\alpha_{x'y'}$ of e_i with endpoints x' and y' is contained in $R_h(\mathbf{s}_i, \mathcal{S})$. Therefore, $\gamma_{xx'} \cup \alpha_{x'y'} \cup \gamma_{yy'}$ contains a path between x and y that is contained in $R_h(\mathbf{s}_i, \mathcal{S})$. □

Lemma 4.8. *Let x be a point in $\mathbb{R}^2$. There is a site $\mathbf{s}_i \in \mathcal{S}$ such that $x \in R_h(\mathbf{s}_i, \mathcal{S})$.*

Proof. Let $\mathcal{S}' \subseteq \mathcal{S}$ be the set of sites $\mathbf{s}_i$ such that $d_h(x, \mathbf{s}_i) = \min_{\mathbf{s}_j \in \mathcal{S}} d_h(x, \mathbf{s}_j)$. Since the hitting distance function is well-defined, this set is non-empty. Let $\mathbf{s}_i$ be the minimum element of $\mathcal{S}'$ according to the total ordering defined on the sites in $\mathcal{S}$. Then, by definition, $x \in R_h(\mathbf{s}_i, \mathcal{S})$. □

Lemma 4.9. *Let $\mathcal{CH}(\mathbf{s}_i)$ and $\mathcal{CH}(\mathbf{s}_j)$ be the convex hulls of the elements of sites $\mathbf{s}_i$ and $\mathbf{s}_j$, respectively. Then $\operatorname{int} \mathcal{CH}(\mathbf{s}_i) \cap \mathcal{CH}(\mathbf{s}_j) = \emptyset$.*

Proof. Assume that $\operatorname{int}\mathcal{CH}(\mathbf{s}_i) \cap \mathcal{CH}(\mathbf{s}_j) \neq \emptyset$. Then at least one element of $\mathbf{s}_i$ is a line segment, or else $\mathcal{CH}(\mathbf{s}_i)$ does not have an interior. Let x be a point in $\operatorname{int}\mathcal{CH}(\mathbf{s}_i) \cap \mathcal{CH}(\mathbf{s}_j)$. We may assume that $x \in \partial\mathcal{P}$ since, by Lemma 4.2, the edges of $\mathcal{CH}(\mathbf{s}_j)$ that are not part of $\partial\mathcal{P}$ must be chords in $\mathcal{P}$. Since $\mathcal{CH}(\mathbf{s}_i) \subseteq \bigcup_{z \in e_i} \mathcal{D}_h(z, \mathbf{s}_i)$ there is a point z on e_i such that $x \in \operatorname{int}\mathcal{D}_h(z, \mathbf{s}_i)$. This is a contradiction since, by Lemma 4.1, $\mathcal{D}_h(z, \mathbf{s}_i)$ cannot contain any part of $\partial\mathcal{P}$ in its interior. Therefore, $\operatorname{int}\mathcal{CH}(\mathbf{s}_i) \cap \mathcal{CH}(\mathbf{s}_j) = \emptyset$. $\square$

Lemma 4.10. *$R_h(\mathbf{s}_i, \mathbf{s}_j)$ is unbounded.*

Proof. Lemma 4.9 implies that there exists a line L that separates the interior of $\mathcal{CH}(\mathbf{s}_i)$ from the interior of $\mathcal{CH}(\mathbf{s}_j)$. Let H_i and H_j be the open half-planes on either side of L such that $\mathcal{CH}(\mathbf{s}_i) \subseteq H_i \cup L$ and $\mathcal{CH}(\mathbf{s}_j) \subseteq H_j \cup L$. Without loss of generality, assume that L is a vertical line passing through the origin, H_i lies to the left of L, and H_j lies to the right of L. There are two cases to consider, based on whether or not H_i is empty.

- H_i contains a point p in an element of $\mathbf{s}_i$. Without loss of generality, assume that $p \in I_L(e_i)$. Let q be a point in $I_R(e_i)$ and let q' be the horizontal projection of q onto L. We may assume that $\mathbf{s}_j$ is not a single-vertex site with reflex vertex q' since situation is covered in the following case. Starting from q', move a point z left along the horizontal line through q' until $\mathcal{D}(z, d(z, q'))$ contains both p and q. Consider any point z' on the horizontal line through q' lying to the left of z. By Lemma 2.5, $\mathcal{D}(z, d(z, q')) \subseteq \mathcal{D}(z', d(z', q'))$, so $d_h(z', \mathbf{s}_i) \leq d(z', q')$. Refer to Figure 4.16. Since $\mathcal{D}(z', d(z', q'))$ does not intersect H_j and touches L at a single point q', $z' \in R_h(\mathbf{s}_i, \mathbf{s}_j)$. Since z' can be placed arbitrarily far away from z, and hence from $\mathbf{s}_i$, along the line through q', $R_h(\mathbf{s}_i, \mathbf{s}_j)$ is unbounded.

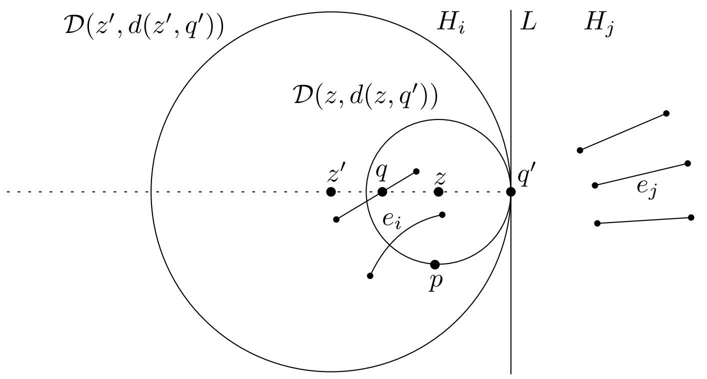


Figure 4.16: $R_h(\mathbf{s}_i, \mathbf{s}_j)$ is unbounded when H_i contains a point p in an element of $\mathbf{s}_i$.

- H_i does not contain any point in an element of $\mathbf{s}_i$. Then $\mathbf{s}_i$ is either a single-vertex site or a vertex-vertex site. Let $p_R \in I_R(e_i)$ and $p_L \in I_L(e_i)$.

  If $\mathbf{s}_i$ is a single-vertex site let $p = p_R = p_L$ and let z' be a point lying to the left of L on the horizontal line through p. Then p is the only point of L in $\mathcal{D}_h(z', \mathbf{s}_i) = \mathcal{D}(z', d(z, p))$. Refer to Figure 4.17. $\mathcal{D}_h(z', \mathbf{s}_i)$ cannot intersect both elements of $\mathbf{s}_j$, since $\mathbf{s}_j$ cannot be a single-vertex site with vertex p, so $z' \in R_h(\mathbf{s}_i, \mathbf{s}_j)$. We may place z' arbitrarily far away from p along the horizontal line through p, so $R_h(\mathbf{s}_i, \mathbf{s}_j)$ is unbounded.

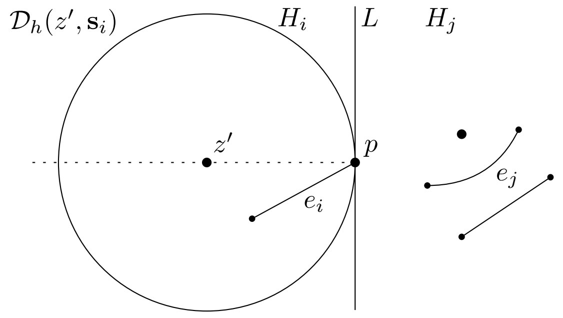


Figure 4.17: $R_h(\mathbf{s}_i, \mathbf{s}_j)$ is unbounded when $\mathbf{s}_i$ is a single-vertex site and H_i does not contain part of an element of $\mathbf{s}_i$.

  If $\mathbf{s}_i$ is a vertex-vertex site we may assume that $p_R \in I_R(e_j)$ and $p_L \in I_L(e_j)$, otherwise we could have chosen L so that either p_R or p_L is in H_i. At least one element of $\mathbf{s}_j$ must be a line segment, otherwise $\mathbf{s}_j = \mathbf{s}_i$, so $\mathbf{s}_i \prec \mathbf{s}_j$. Let z be a point in $\operatorname{int} e_i$. By Lemma 4.6, $z \in R_h(\mathbf{s}_i, \mathbf{s}_j)$. Let z' be any point on the horizontal line through z that lies to the left of z. By Lemma 2.6, $\mathcal{D}(z', \mathbf{s}_i) \cap H_j \subseteq \mathcal{D}_h(z, \mathbf{s}_i)$, so $z' \in R_h(\mathbf{s}_i, \mathbf{s}_j)$. Refer to Figure 4.18. Again, since we may place z' arbitrarily far away from z along the horizontal line through z, $R_h(\mathbf{s}_i, \mathbf{s}_j)$ is unbounded. □

Lemma 4.11. *$B^\star_h(\mathbf{s}_i, \mathbf{s}_j)$ is homeomorphic to a line.*

Proof. $B^\star_h(\mathbf{s}_i, \mathbf{s}_j)$ is non-empty and does not contain any 2-dimensional region, due to the total ordering defined on $\mathcal{S}$. By Lemmas 4.7 and 4.10, the Voronoi regions $R_h(\mathbf{s}_i, \mathbf{s}_j)$ and $R_h(\mathbf{s}_j, \mathbf{s}_i)$ are path-connected and unbounded, implying that $B^\star_h(\mathbf{s}_i, \mathbf{s}_j)$ is acyclic and unbounded. This is sufficient to prove that $B^\star_h(\mathbf{s}_i, \mathbf{s}_j)$ is homeomorphic to a line [7]. □

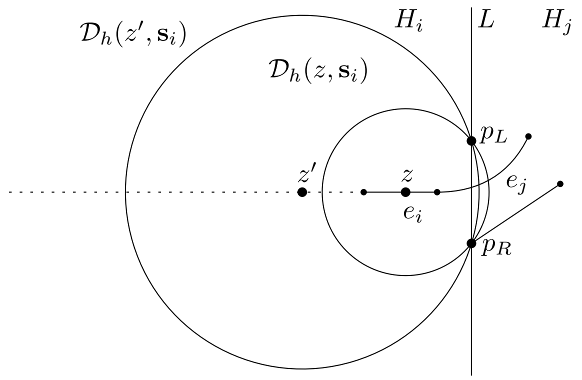

$\mathcal{D}_h(z', \mathbf{s}_i)$
H_i
L
H_j
$\mathcal{D}_h(z, \mathbf{s}_i)$
p_L
z'
z
e_j
e_i
p_R

Figure 4.18: Proof of Lemma 4.10 when $\mathbf{s}_i$ is a vertex-vertex site and H_i does not contain part of an element of $\mathbf{s}_i$.

Lemma 4.12. *$B^\star_h(\mathbf{s}_i, \mathbf{s}_j)$ has constant complexity.*

Proof. The hitting distance bisector of two sites consists of pieces of the boundaries of the Euclidean bisectors between elements of $\mathbf{s}_i$ and $\mathbf{s}_j$. Each site consists of at most two reflex vertices or line segments, so $B^\star_h(\mathbf{s}_i, \mathbf{s}_j)$ consists of a constant number of arcs of lines and parabolas. □

Lemma 4.13. *The intersection of two chosen hitting distance bisectors consists of a constant number of connected components.*

Proof. This follows directly from Lemma 4.12. Each chosen hitting distance bisector consists of a constant number of arcs of lines and parabolas so the intersection of two chosen hitting distance bisectors consists of a constant number of connected components. □

Lemma 4.14. *$\mathcal{V}_h(\mathcal{S})$ is an abstract Voronoi diagram.*

Proof. By Lemma 4.11 the chosen hitting distance bisectors of the sites in $\mathcal{S}$ are homeomorphic to lines and by Lemma 4.13 the intersection of any two of these bisectors consists of a constant number of connected components. By Lemma 4.7 the Voronoi regions of $\mathcal{S}$ are path-connected and by Lemma 4.8 the union of these regions covers $\mathbb{R}^2$. Therefore, $\mathcal{V}_h(\mathcal{S})$ is an abstract Voronoi diagram [31]. □

Since $\mathcal{V}_h(\mathcal{S})$ is an abstract Voronoi diagram, it has size $O(k)$. There are several optimal algorithms for constructing abstract Voronoi diagrams, including an $O(k \log k)$ divide and conquer algorithm by Klein [30] and an $O(k \log k)$ expected time randomized incremental construction algorithm by Klein et al. [32]. The randomized incremental construction algorithm is of particular interest since it only requires that we are able to construct the hitting distance Voronoi diagram of five sites in constant time. The chosen hitting distance bisector between any pair of sites has constant complexity so the diagram of five sites can be computed explicitly from these bisectors in constant time.

After constructing the hitting distance Voronoi diagram of $\mathcal{S}$ in $O(k \log k)$ time we can construct an $O(k)$ size point location data structure on $\mathcal{V}_h(\mathcal{S})$ in $O(k \log k)$ time so that the site closest to a given point x can be found in $O(\log k)$ time [39, 44, 8]. We conclude with the following theorem.

Theorem 4.15. *A set $\mathcal{S}$ of k sites associated with k edges in $\mathcal{V}(\mathcal{P})$ can be preprocessed in $O(k \log k)$ time into a data structure of size $O(k)$ so that, given a point $x \in \mathbb{R}^2$, a site $\mathbf{s}_i \in \mathcal{S}$ such that $d_h(x, \mathbf{s}_i) = \min_{\mathbf{s}_j \in \mathcal{S}} d_h(x, \mathbf{s}_j)$ can be found in $O(\log k)$ time.*

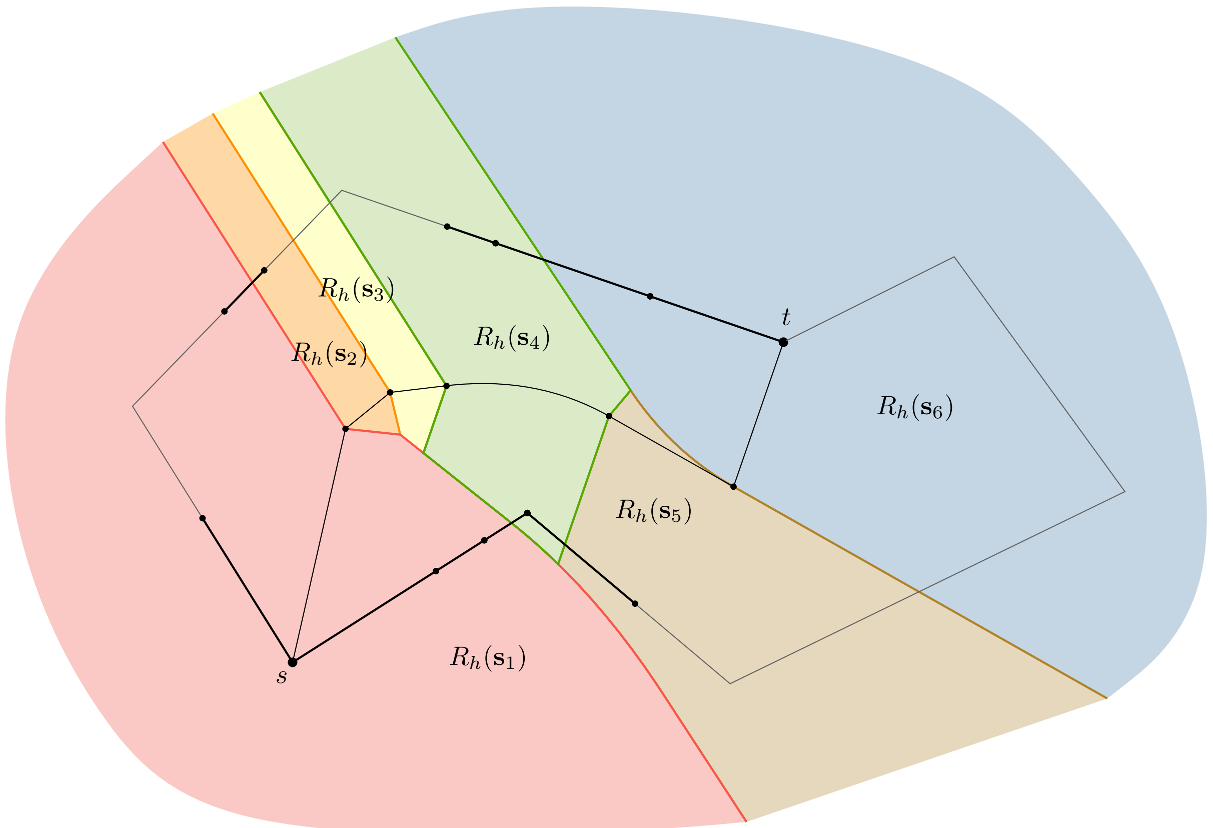


Figure 4.19: The hitting distance Voronoi diagram of the sites corresponding to the edges of a path between s and t in $\mathcal{V}(\mathcal{P})$.

4.4 Obstacle-avoiding path existence queries

In this section, we show how to construct data structures so that, given two arbitrary vertices s and t of $\mathcal{P}$, a site corresponding to an edge of a path between s and t in $\mathcal{V}(\mathcal{P})$ that is closest to the center c of a query disk $\mathcal{D}(c, r)$ under the hitting distance can be found efficiently.

The simplest approach is to construct hitting distance Voronoi diagrams for the paths between every pair of vertices in $\mathcal{P}$. By Theorem 4.15, this can be done in $O(n^2 \log n)$ time and results in a data structure of size $O(n^2)$ that supports $O(\log n)$ time closest site queries. By Corollary 4.4, this data structure can be used to answer obstacle-avoiding path existence queries in $O(\log n)$ time.

Theorem 4.16. *A simple polygon $\mathcal{P}$ with n vertices can be preprocessed in $O(n^2 \log n)$ time into a data structure of size $O(n^2)$ so that, given two points s and t in $\mathcal{P}$ and disk obstacle $\mathcal{O}$, it is possible to determine whether there is an obstacle-avoiding path between s and t in $O(\log n)$ time.*

We now show how to reduce the preprocessing time and space requirements, albeit at the expense of query time. The general idea is to decompose $\mathcal{V}(\mathcal{P})$ into $O(n)$ paths so that any path between any pair of vertices s and t in $\mathcal{V}(\mathcal{P})$ is contained in $O(\log n)$ of the paths in the decomposition. For each subpath in the decomposition, we store a range tree on the edges of that subpath, and for each node in each range tree we construct a hitting distance Voronoi diagram on the sites corresponding to the edges of the subpath represented by the subtree of that node. Using this data structure, we can find a site that is closest to the center c of a query disk $\mathcal{D}(c, r)$, under the hitting distance, in $O(\log^3 n)$ time.

The subpath decomposition of $\mathcal{V}(\mathcal{P})$ is based on the following theorem, first introduced by Cole and Vishkin to evaluate expression trees in parallel [17], and later used by Narasimhan and Smid for lowest common ancestor queries in trees [40].

Theorem 4.17 ([17, 40])**.** *Let T be a tree of size m and let $\mathrm{size}(v)$ denote the size of the subtree rooted at a node v in T. Let u be a node in T and let $v_1, \ldots, v_\ell$ be the children of u. There is at most one child v_i, $1 \le i \le \ell$, such that $\lfloor \log(\mathrm{size}(v_i)) \rfloor = \lfloor \log(\mathrm{size}(u)) \rfloor$; for every other child v_j, $1 \le j \le \ell$, such that $v_j \ne v_i$, $\lfloor \log(\mathrm{size}(v_j)) \rfloor < \lfloor \log(\mathrm{size}(u)) \rfloor$.*

We will assume that $\mathcal{V}(\mathcal{P})$ is rooted at an arbitrary vertex v_r in $\mathcal{V}(\mathcal{P})$. By Theorem 4.17, for every vertex v in $\mathcal{V}(\mathcal{P})$ there is a unique maximal sequence $v_1, \ldots, v_k$ of vertices of $\mathcal{V}(\mathcal{P})$, which we call a *group*, that contains v and has the following properties.

- $\lfloor \log(\text{size}(v_i)) \rfloor = \lfloor \log(\text{size}(v_j)) \rfloor$ for all $1 \le i < j \le k$; and
- $\text{par}(v_i) = v_{i+1}$ for all $1 \le i < k$, where $\text{par}(v_i)$ denotes the parent of v_i in $\mathcal{V}(\mathcal{P})$.

Let $e_1, \ldots, e_k$ be the edges of $\mathcal{V}(\mathcal{P})$ such that each edge e_i, $1 \le i \le k$, has endpoints v_i and $\text{par}(v_i)$. Note that one vertex of e_k is not part of the group $v_1, \ldots, v_k$, and if $v_k = v_r$ then there is no edge e_k at all. For simplicity, we will assume that $v_k \neq v_r$. Construct a range tree T on the edges $e_1, \ldots, e_k$. For each node x in T, construct a hitting distance Voronoi diagram on the subset of the edges $e_1, \ldots, e_k$ stored in the subtree rooted at x. By Theorem 4.15, the range tree has size $O(k \log k)$ can be constructed in $O(k \log^2 k)$ time. We store this range tree at v_k, the vertex in the sequence that is closest to the root, and with each node v_i, $1 \le i \le k$, we store a *group parent* pointer to v_k, $\text{gpar}(v_i) = v_k$. Let $T[v_i, v_j]$, $1 \le i < j \le k$ denote the $O(\log k)$ hitting distance Voronoi diagrams of the sites associated with the edges $e_i, \ldots, e_j$. These diagrams can be found in $O(\log k)$ time using the range tree T stored at v_k.

By Theorem 4.17, each vertex of $\mathcal{V}(\mathcal{P})$ belongs to exactly one group and there are $O(n)$ groups in total. If we construct a range tree for each of these groups then the size of the resulting data structure is $O(n \log n)$ and it can be computed in $O(n \log^2 n)$ time. Figure 4.20 shows part of a path decomposition of a Voronoi diagram $\mathcal{V}(\mathcal{P})$. In this figure, the thick edges represent the edges of a path between s and t in $\mathcal{V}(\mathcal{P})$. The vertices of this path are contained in the groups $G_1, \ldots, G_6$. Notice that there is only one edge of G_3 in the path between s and t, and this edge is in the middle of the group. This illustrates why we need to make use of the range tree data structure.

We now describe how to perform an obstacle-avoiding path existence query. Let s and t be vertices of $\mathcal{P}$ and let $\mathcal{O} = \mathcal{D}(c, r)$ be a disk obstacle. Let v_{lca} be the lowest common ancestor of s and t in $\mathcal{V}(\mathcal{P})$. The general idea is to walk up $\mathcal{V}(\mathcal{P})$ from s to v_{lca} and from t to v_{lca} by following the group parent pointers stored at each node. For each group encountered when walking towards v_{lca} we search the range tree stored with the group parent of that group in $O(\log^2 n)$ time to find a site $\mathbf{s}$ in a hitting distance Voronoi diagrams stored in that tree that is closest to c under the hitting distance. If $d_h(c, \mathbf{s}) \le r$ then, by Lemma 4.3, we can report that there is no obstacle-avoiding path. Otherwise, we continue walking towards v_{lca} by following the parent pointers of the group parent pointers to find the next group on the path from s to v_{lca}. If we reach v_{lca} from s and from t without finding a site $\mathbf{s}$ such that $d_h(c, \mathbf{s}) \le r$ then, by Corollary 4.4, there is an obstacle-avoiding path between s and t. Since there are $O(\log n)$ groups between s and v_{lca} we search $O(\log n)$ range trees for a total query time of $O(\log^3 n)$. We present this algorithm in more detail in Algorithm 4.1. We conclude with the following theorem.

Theorem 4.18. *A simple polygon $\mathcal{P}$ with n vertices can be preprocessed in $O(n \log^2 n)$ time into a data structure of size $O(n \log n)$ so that, given two points s and t in $\mathcal{P}$, and disk obstacle $\mathcal{O}$, it is possible to determine whether there is an obstacle-avoiding path between s and t in $O(\log^3 n)$ time.*

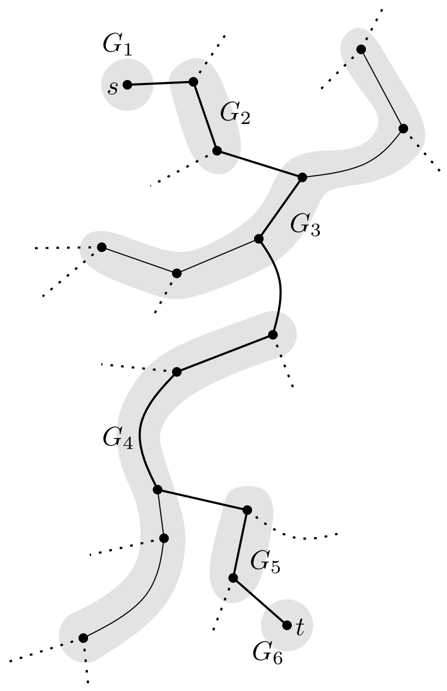


Figure 4.20: Groups $G_1, \ldots, G_6$ contain the vertices of a path between s and t in $\mathcal{V}(\mathcal{P})$.

Algorithm 4.1: Single disk obstacle-avoiding path existence query.

```
Input: Vertices $s$ and $t$ of $\mathcal{P}$ and a disk obstacle $\mathcal{O} = \mathcal{D}(c, r)$
Output: Returns whether there is an obstacle-avoiding path between $s$ and $t$.
$v_{lca} \leftarrow$ lowest common ancestor of $s$ and $t$ in $\mathcal{V}(\mathcal{P})$
for $v \in \{s, t\}$ do
    while gpar($v$) $\neq$ gpar($v_{lca}$) do
        $T \leftarrow$ the range tree stored at gpar($v$)
        $X \leftarrow T[v, \mathrm{par}(\mathrm{gpar}(v))]$
        for $\mathcal{V}_h(\mathcal{S}) \in X$ do
            $\mathbf{s}_i \leftarrow$ a site in $\mathcal{S}$ that is closest to $c$ under the hitting distance
            if $d_h(c, \mathbf{s}_i) \leq r$ then
                return false
            end
        end
        $v \leftarrow$ par(gpar($v$))
    end
    if $v \neq v_{lca}$ then
        $T \leftarrow$ the range tree stored at gpar($v_{lca}$).
        $X \leftarrow T[v, v_{lca}]$
        for $\mathcal{V}_h(\mathcal{S}) \in X$ do
            $\mathbf{s}_i \leftarrow$ a site in $\mathcal{S}$ that is closest to $c$ under the hitting distance
            if $d_h(c, \mathbf{s}_i) \leq r$ then
                return false
            end
        end
    end
end
return true
```

Chapter 5

Conclusion

In this thesis, we showed how to preprocess a simple polygon $\mathcal{P}$ so that, given two points s and t inside $\mathcal{P}$, and a set $\mathcal{Q}$ of h pairwise disjoint obstacles that are either convex polygons or disks, it is possible to determine whether there is an obstacle-avoiding path between s and t.

5.1 Contributions

In Chapter 3 we focused on sets of pairwise disjoint convex polygon obstacles. We made use of the shortest path data structure of Guibas and Hershberger [22] to find the connected components of the intersection between $\pi(s,t)$ and each obstacle in $\mathcal{Q}$. In order to do so, we augmented each precomputed hourglass stored in the shortest path data structure with a ray shooting data structure. We showed that the augmented shortest path data structure has size $O(n \log n)$ and can be constructed in $O(n \log n)$ time. We then proved that there is an obstacle-avoiding path between s and t if and only if, for each connected component c of $\pi(s,t) \cap \bigcup_{\mathcal{O}_i \in \mathcal{Q}} \mathcal{O}_i$, it is possible to walk clockwise or counter-clockwise along the boundary of the obstacle $\mathcal{O}_i$ containing c, from one endpoint of c to the other, without hitting the boundary of $\mathcal{P}$. In order to obtain query times sublinear in the size of $\mathcal{P}$ we could not consider every connected component in the intersection of $\pi(s,t)$ and the obstacles in $\mathcal{Q}$, since there could be $O(n)$ components in total. To get around this, we showed that if the intersection of $\pi(s,t)$ with an edge of an obstacle in $\mathcal{Q}$ consists of more than two connected components then there cannot be an obstacle-avoiding path between s and t. This gives us an upper bound of $2m$ on the number of connected components of $\pi(s,t) \cap \bigcup_{\mathcal{O}_i \in \mathcal{Q}} \mathcal{O}_i$ that need to be checked before we can determine whether there is an obstacle-avoiding path between s and t, where m is the sum of the number of edges in each obstacle in $\mathcal{Q}$. Since we only need to check if we can walk around $O(m)$ of the connected components, we obtain $O(m \log^2 n)$ time obstacle-avoiding path existence queries.

In Chapter 4 we considered sets of pairwise disjoint disk obstacles. We started by reviewing the Euclidean Voronoi diagram of a simple polygon. In particular, we noted that we may consider this diagram to be a tree whose leaves are the vertices of $\mathcal{P}$, and that each edge in the diagram is defined by two pieces of the boundary of $\mathcal{P}$ that are either reflex vertices or subsegments of the edges of $\mathcal{P}$. In order to solve obstacle-avoiding path existence queries, we introduced the hitting distance and showed that, by considering the pieces of the boundary of $\mathcal{P}$ defining each edge of the Voronoi path between s and t, the hitting distance can be used to determine whether there is an obstacle-avoiding path between s and t. Specifically, we proved that there is an obstacle-avoiding path between s and t if and only if there is an edge in the Voronoi path between s and t such that the hitting distance between the center of each disk obstacle $\mathcal{O}_i$ in $\mathcal{Q}$ and the set containing the pieces of the boundary defining that edge is less than or equal to the radius of $\mathcal{O}_i$. We then showed that the Voronoi digram of these edge sites under the hitting distance can be constructed efficiently by proving that the hitting distance Voronoi diagram is an abstract Voronoi diagram. Finally, we showed how to construct data structures containing hitting distance Voronoi diagrams so that obstacle-avoiding path existence queries can be solved in $O(h \log n)$ time using a data structure of size $O(n^2)$ that can be constructed in $O(n^2 \log n)$ time, or in $O(h \log^3 n)$ time using a data structure of size $O(n \log n)$ that can be constructed in $O(n \log^2 n)$ time, where h is the number of disk obstacles in the query.

5.2 Future work

There are several possible directions for future work, in addition to the obvious suggestions of improving the time and space complexities of the obstacle-avoiding path existence algorithms and data structures described in this thesis.

Obstacle-avoiding path existence queries

In this thesis, we considered obstacle-avoiding path existence queries containing sets of disjoint convex polygon and disk obstacles. The solutions we presented are likely non-optimal. What are the lower bounds for these obstacle-avoiding path existence queries?

A natural question to ask is, for what other types of obstacles can we solve obstacle-avoiding path existence queries efficiently? If the obstacles are all translated copies of a fixed convex polygon, or disks of the same radius, can we improve the running time of the path-existence queries?

If the input domain is a polygon with holes, can we solve obstacle-avoiding path existence queries efficiently? In this setting, we cannot consider each obstacle independently, as it is possible for the union of set of disjoint obstacles to block the path between s and t even if every obstacle individually does not, so in this setting it makes sense to start by considering queries containing a single obstacle.

Shortest obstacle-avoiding paths

Another direction is to find and report shortest obstacle-avoiding paths if such paths exist. If the obstacles are convex polygons, it might be possible to use the shortest path data structure of Guibas and Hershberger [22] to find a shortest path between the query points. This problem appears to be more difficult if the obstacles are disks.

If it is possible report a shortest obstacle-avoiding path under the Euclidean metric efficiently then it makes sense to consider other distance functions, such as the L_1 metric [14, 13], convex distance functions [25, 9, 20], or the link-distance, where the link-distance between s and t is the minimum number of turns required to construct a path between s and t [46, 3, 38].

Chen notes in his recent survey on geometric shortest path queries that very little research has focused on two-point shortest path queries in dynamic environments [12], so there are many open problems in this area. We refer the reader to the open problems listed at the end of his survey for more ideas on potential areas of research.

Hitting distance Voronoi diagrams

In Section 4.3 we noted the similarities between the Hausdorff Voronoi diagram and the hitting distance Voronoi diagram and mentioned that the Hausdorff Voronoi diagram of a set of pairwise non-crossing sites is an abstract Voronoi diagrams. Our analysis of the hitting distance Voronoi diagram was restricted to a very specific set of sites. It would be interesting to see if the hitting distance Voronoi diagram of pairwise non-crossing sites is also an abstract Voronoi diagram.

The closest covered set diagram of Abellanas et al. is an abstract Voronoi diagram when the Hausdorff distance is defined using any convex distance function [1]. If we define the hitting distance to a site as

$$d_h(x, \mathbf{s}) = \max_{Y \in \mathbf{S}} \min_{y \in Y} \; d_c(x, y),$$

where d_c is a convex distance function, is the hitting distance Voronoi diagram still an abstract Voronoi diagram? If this is the case, then we can solve obstacle-avoiding path existence queries for sets of obstacles that are homothetic copy of the compact convex subset defining d_c.

Bibliography

[1] Manuel Abellanas, Gregorio Hernández-Peñalver, Rolf Klein, Victor Neumann-Lara, and Jorge Urrutia. A combinatorial property of convex sets. *Discrete & Computational Geometry*, 17(3):307–318, 1997.

[2] Oswin Aichholzer, Franz Aurenhammer, and Belén Palop. Quickest paths, straight skeletons, and the city Voronoi diagram. *Discrete & Computational Geometry*, 31(1):17–35, 2004.

[3] Esther M. Arkin, Joseph S. B. Mitchell, and Subhash Suri. Logarithmic-time link path queries in a simple polygon. *Int. J. Comput. Geometry Appl.*, 5(4):369–395, 1995.

[4] Boris Aronov. On the geodesic Voronoi diagram of point sites in a simple polygon. *Algorithmica*, 4(1):109–140, 1989.

[5] Franz Aurenhammer and Rolf Klein. Voronoi diagrams. *Handbook of Computational Geometry*, 5:201–290, 2000.

[6] Franz Aurenhammer, Rolf Klein, and Der-Tsai Lee. *Voronoi Diagrams and Delaunay Triangulations*. World Scientific, 2013.

[7] Sang Won Bae and Kyung-Yong Chwa. Voronoi diagrams for a transportation network on the Euclidean plane. *Int. J. Comput. Geometry Appl.*, 16(2-3):117–144, 2006.

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

[9] Sergei Bespamyatnikh. Computing homotopic shortest paths in the plane. *J. Algorithms*, 49(2):284–303, 2003.

[10] Bernard Chazelle. A theorem on polygon cutting with applications. In *FOCS*, pages 339–349. IEEE Computer Society, 1982.

[11] Bernard Chazelle. Triangulating a simple polygon in linear time. *Discrete & Computational Geometry*, 6:485–524, 1991.

[12] Danny Z. Chen. Efficient algorithms for geometric shortest path query problems. In Panos M. Pardalos, Ding-Zhu Du, and Ronald L. Graham, editors, *Handbook of Combinatorial Optimization*, pages 1125–1154. Springer New York, 2013.

[13] Danny Z. Chen, Rajasekhar Inkulu, and Haitao Wang. Two-point L1 shortest path queries in the plane. In Siu-Wing Cheng and Olivier Devillers, editors, *Symposium on Computational Geometry*, page 406. ACM, 2014.

[14] Danny Z. Chen, Kevin S. Klenk, and Hung-Yi Tu. Shortest path queries among weighted obstacles in the rectilinear plane. *SIAM J. Comput.*, 29(4):1223–1246, 2000.

[15] Yi-Jen Chiang and Joseph S. B. Mitchell. Two-point euclidean shortest path queries in the plane. In Robert Endre Tarjan and Tandy Warnow, editors, *SODA*, pages 215–224. ACM/SIAM, 1999.

[16] Francis Y. L. Chin, Jack Snoeyink, and Cao An Wang. Finding the medial axis of a simple polygon in linear time. *Discrete & Computational Geometry*, 21(3):405–420, 1999.

[17] Richard Cole and Uzi Vishkin. The accelerated centroid decomposition technique for optimal parallel tree evaluation in logarithmic time. *Algorithmica*, 3:329–346, 1988.

[18] Frank Dehne, Anil Maheshwari, and Ryan Taylor. A coarse grained parallel algorithm for Hausdorff Voronoi diagrams. In *ICPP*, pages 497–504. IEEE Computer Society, 2006.

[19] Herbert Edelsbrunner, Leonidas J. Guibas, and Micha Sharir. The upper envelope of piecewise linear functions: Algorithms and applications. *Discrete & Computational Geometry*, 4:311–336, 1989.

[20] Alon Efrat, Stephen G. Kobourov, and Anna Lubiw. Computing homotopic shortest paths efficiently. *Comput. Geom.*, 35(3):162–172, 2006.

[21] Michael T. Goodrich and Roberto Tamassia. Dynamic ray shooting and shortest paths in planar subdivisions via balanced geodesic triangulations. *J. Algorithms*, 23(1):51–73, 1997.

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

[23] Hua Guo, Anil Maheshwari, and Jörg-Rüdiger Sack. Shortest path queries in polygonal domains. In Rudolf Fleischer and Jinhui Xu, editors, *AAIM*, volume 5034 of *Lecture Notes in Computer Science*, pages 200–211. Springer, 2008.

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

[25] John Hershberger and Jack Snoeyink. Computing minimum length paths of a given homotopy class. *Comput. Geom.*, 4:63–97, 1994.

[26] John Hershberger and Subhash Suri. A pedestrian approach to ray shooting: Shoot a ray, take a walk. *J. Algorithms*, 18(3):403–431, 1995.

[27] John Hershberger and Subhash Suri. An optimal algorithm for euclidean shortest paths in the plane. *SIAM J. Comput.*, 28(6):2215–2256, 1999.

[28] Christian Icking, Rolf Klein, Lihong Ma, Stefan Nickel, and Ansgar Weißler. On bisectors for different distance functions. *Discrete Applied Mathematics*, 109(1-2):139–161, 2001.

[29] Yi Jen Chiang, Franco P. Preparata, and Roberto Tamassia. A unified approach to dynamic point location, ray shooting, and shortest paths in planar maps. *SIAM J. Comput.*, 25(1):207–233, 1996.

[30] Rolf Klein. *Concrete and Abstract Voronoi Diagrams*, volume 400 of *Lecture Notes in Computer Science*. Springer, 1989.

[31] Rolf Klein, Elmar Langetepe, and Zahra Nilforoushan. Abstract Voronoi diagrams revisited. *Comput. Geom.*, 42(9):885–902, 2009.

[32] Rolf Klein, Kurt Mehlhorn, and Stefan Meiser. Randomized incremental construction of abstract Voronoi diagrams. *Comput. Geom.*, 3:157–184, 1993.

[33] Steven M. LaValle. *Planning algorithms*. Cambridge University Press, 2006.

[34] Der-Tsai Lee. Medial axis transformation of a planar shape. *IEEE Trans. Pattern Anal. Mach. Intell.*, 4(4):363–369, 1982.

[35] Der-Tsai Lee and Franco P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks*, 14(3):393–410, 1984.

[36] Lihong Ma. *Bisectors and Voronoi diagrams for convex distance functions.* PhD thesis, Fachbereich Informatik, FernUniversität Hagen, 2000.

[37] Joseph S. B. Mitchell. Shortest paths and networks. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 607–641. CRC Press, Inc., Boca Raton, FL, USA, 2004.

[38] Joseph S. B. Mitchell, Valentin Polishchuk, and Mikko Sysikaski. Minimum-link paths revisited. *Comput. Geom.*, 47(6):651–667, 2014.

[39] Ketan Mulmuley. A fast planar partition algorithm, I. *J. Symb. Comput.*, 10(3/4):253–280, 1990.

[40] Giri Narasimhan and Michiel H. M. Smid. *Geometric Spanner Networks.* Cambridge University Press, 2007.

[41] Evanthia Papadopoulou. Critical area computation for missing material defects in VLSI circuits. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 20(5):583–597, 2001.

[42] Evanthia Papadopoulou and Der-Tsai Lee. The Min-Max Voronoi diagram of polygons and applications in VLSI manufacturing. In Prosenjit Bose and Pat Morin, editors, *ISAAC*, volume 2518 of *Lecture Notes in Computer Science*, pages 511–522. Springer, 2002.

[43] Evanthia Papadopoulou and Der-Tsai Lee. The Hausdorff Voronoi diagram of polygonal objects: a divide and conquer approach. *Int. J. Comput. Geometry Appl.*, 14(6):421–452, 2004.

[44] Raimund Seidel. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Comput. Geom.*, 1:51–64, 1991.

[45] Michael Ian Shamos and Dan Hoey. Closest-point problems. In *FOCS*, pages 151–162. IEEE Computer Society, 1975.

[46] Subhash Suri. A linear time algorithm with minimum link paths inside a simple polygon. *Comput. Vision Graph. Image Process.*, 35(1):99–110, July 1986.