# Farthest Neighbors and Center Points in the Presence of Rectangular Obstacles

Boaz Ben-Moshe[1] Matthew J. Katz[1*] Joseph S. B. Mitchell[2†]

[1]Department of Computer Science
Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel
[2]Department of Applied Mathematics and Statistics
State University of New York, Stony Brook, NY 11794-3600

**Abstract**

We study several natural proximity and facility location problems that arise for a set $\mathcal{P}$ of $n$ points and a set $\mathcal{R}$ of $m$ disjoint rectangular obstacles in the plane, where distances are measured according to the $L_1$ shortest path (geodesic) metric. In particular, we compute, in time $O(mn\log(m+n))$, a data structure of size $O(mn)$ that supports $O(\log(m+n))$-time farthest point queries; we avoid computing the more complicated farthest neighbor Voronoi diagram, whose combinatorial complexity we show to be $\Theta(mn)$. We study the center point problem, finding in $O(mn\log(m+n))$ time a center point (and the set of center points) that minimize the maximum distance to sites of $\mathcal{P}$; this result improves the best previous bound by a factor of roughly $m$. In addition, we give algorithms for approximating the diameter, $D$, and radius, $r$, of $\mathcal{P}$, including methods to (i) compute a pair of points $a, b \in \mathcal{P}$, such that $d(a,b) \geq (1-\varepsilon)D$, in $O(n\log n + \frac{1}{\varepsilon}(n+m)\log m)$ time; and (ii) compute a point $c'$, such that $\max\{d(p,c') \mid p \in \mathcal{P}\} \leq (1+\varepsilon)r$, in $O(n\log(m+n) + (m/\varepsilon)\log(m+1/\varepsilon))$ time. Finally, we show that for all the problems above it is enough to consider only a subset of $\mathcal{P}$. This subset is likely to be much smaller than $\mathcal{P}$, it is computable in $O(n\log n)$ time, and using it results in significantly decreased runtime in practice.

## 1 Introduction

Let $\mathcal{R}$ be a set of $m$ disjoint (axis-aligned) rectangles in the plane. We assume the rectangles in $\mathcal{R}$ are open, and refer to them as *obstacles*. Let $\mathcal{P}$ be a set of $n$ points in the *free space*, $\mathcal{F} = \mathbb{R}^2 - \cup\mathcal{R}$. In all of the problems that are studied in this paper the input is the pair $\mathcal{R}$, $\mathcal{P}$, and the distance between two points $a, b \in \mathcal{F}$, denoted $d(a,b)$, is the $L_1$ geodesic (shortest obstacle-avoiding path) distance between them; i.e., $d(a,b)$ is the (Euclidean) length of a shortest path in $\mathcal{F}$, from $a$ to $b$, that is comprised of axis-parallel line segments.

In this paper, we give new results on several fundamental proximity problems, including farthest neighbor queries, center point, diameter, and radius of $\mathcal{P}$.

*Supported by an Intel research grant.

†Supported in part by HRL Laboratories, the National Science Foundation (CCR-9732221), NASA Ames Research Center, Northrop-Grumman Corporation, Sandia National Labs, Seagull Technology, and Sun Microsystems.

The first problem we address is that of constructing a data structure for farthest point queries in $\mathcal{P}$, which enables one to determine quickly which point of $\mathcal{P}$ is farthest from a query point $q$. We note that the *nearest* neighbor query problem for $L_1$ geodesic distance in the plane is well understood: one can compute the relevant Voronoi diagram, of size $O(m+n)$, in time $O((m+n)\log(m+n))$, and then perform point location queries in time $O(\log(m+n))$ [6, 11, 12]. In contrast, we are not aware of prior results on $L_1$ geodesic farthest neighbor Voronoi diagrams. We show that the combinatorial complexity of the $L_1$ farthest neighbor Voronoi diagram is $\Omega(mn)$. This bound is somewhat surprising given that the complexity of the farthest neighbor $L_1$ Voronoi diagram if there are *no* obstacles is $O(1)$, and the complexity of the $L_1$ nearest neighbor Voronoi diagram is $O(m+n)$. For our problem, though, we avoid computing the $L_1$ farthest neighbor Voronoi diagram, which seems to be more difficult; indeed, we do not know of an algorithm to compute the diagram in time close to its size. Instead, we present a simpler data structure of size $O(mn)$ that enables us to answer a farthest neighbor query in $O(\log(m+n))$ time. The construction time is $O(mn\log(m+n))$.

Using our data structure, we can compute all farthest neighbors in $O(mn\log(m+n))$ time. Previously, the most efficient way to do this was to construct the $O(m^2)$-size data structure of Atallah and Chen [3], which enables $O(\log m)$-time distance queries between a pair of points in $\mathcal{F}$; doing this for all $O(n^2)$ pairs yields a time bound of $O(m^2+n^2\log m)$.

The second main problem that we study is the *center point problem*: determine a point $c\in\mathcal{F}$ that minimizes the maximum distance, $r=\max\{d(c,p)\mid p\in\mathcal{P}\}$, to all sites of $\mathcal{P}$. We are also interested in computing the *center set* of all center points $c$. Choi et al. [5] studied a weighted version of this problem, in which a positive weight $w(p)$ is associated with each of the points $p\in\mathcal{P}$, and a center point $c$ is a point minimizing $\max\{w(p)d(c,p)\mid p\in\mathcal{P}\}$. They show that the center set consists of a union of $\Theta(mn)$ line segments (each forming an angle of $\pm 45$ degrees with the horizontal) and give an $O(m^2 n\log^2 n)$-time algorithm for computing the center set; for unweighted points, they give an $O(m^2 n)$ algorithm.

In this paper, we present an $O(mn\log(m+n))$ algorithm for computing a single (unweighted) center point. We can compute the center set within the same time bound, thus improving the previous bound by a factor of roughly $m$. It is easy to see that if $c$ is a center point, it must lie on an edge of the $L_1$ farthest neighbor Voronoi diagram. We prove a theorem (Theorem 5.4) that allows us to further restrict our search for a center point. We then employ our data structure for farthest point queries in order to find such a point.

In Section 3 we prove a theorem (Theorem 3.1) which is very significant from a practical point of view. This theorem allows us to ignore most of the points in $\mathcal{P}$ in the solution of both of our problems (after a preprocessing stage of $O(n\log n)$ time). If $k$ is the size of the relevant subset of $\mathcal{P}$, then in the bounds above one can replace $n$ with $k$. Typically, $k$ is roughly $O(\sqrt{n})$ (its expected size, if, e.g., $\mathcal{P}$ is uniformly distributed in a square).

Finally, in Section 6 we present efficient algorithms for approximating the diameter and *radius* of $\mathcal{P}$. More precisely, for any constant $\varepsilon>0$, we present (i) an $O(n\log n+\frac{1}{\varepsilon}(n+m)\log m)$ algorithm that finds a pair of points $a,b\in\mathcal{P}$, such that $d(a,b)\geq(1-\varepsilon)D$, where $D$ is the diameter of $\mathcal{P}$, and (ii) an $O(n\log(m+n)+(m/\varepsilon)\log(m+1/\varepsilon))$ algorithm that finds a point $c'$, such that $\max\{d(p,c')\mid p\in\mathcal{P}\}\leq(1+\varepsilon)r$. These algorithms are based on a slightly stronger version of Theorem 3.1.

**Related Work.** In addition to the prior work cited above from the computational geometry community, there has been quite a bit of attention paid to the problems we study in the facility location and operations research community. Specifically, center problems in the presence of barriers and the $L_1$ metric are studied by Dearing et al [8]. We do not attempt here to survey the field; for extensive background and numerous references, we refer the reader to the recent PhD thesis of Segars [14] and the Habilitation thesis of Klamroth [10].

## 2 Preliminaries

For points $a, b \in \mathcal{F}$, we let $path(a, b)$ denote any path between $a$ and $b$ of length $d(a, b)$. Assume that $a$ lies to the left of (resp., below) $b$. An *$x$-monotone* (resp., *$y$-monotone*) path from $a$ to $b$ is a path in which all of the horizontal (resp., vertical) segments of the path are directed rightwards (resp., upwards).

**Claim 2.1** *[4, 7] For any two points $a$ and $b$ in the free space, any shortest path between $a$ and $b$ is either $x$-monotone or $y$-monotone (or both).*

A path that is both $x$-monotone and $y$-monotone is *$xy$-monotone*. Clearly, if there exists an $xy$-monotone path between $a = (a_x, a_y)$ and $b = (b_x, b_y)$, then this path is a shortest path between $a$ and $b$ and its length is simply $|a_x - b_x| + |a_y - b_y|$. Moreover, in this case *all* shortest paths between $a$ and $b$ must be $xy$-monotone.

**Claim 2.2** *If there exist both an $x$-monotone path (between $a$ and $b$) and a $y$-monotone path, then there exists an $xy$-monotone path between $a$ and $b$.*

**Corollary 2.3** *If there exists an $x$-monotone (resp., $y$-monotone) path between $a$ and $b$, then* all *shortest paths between $a$ and $b$ are $x$-monotone (resp., $y$-monotone).*

Let $a, b \in \mathcal{F}$, with $a$ to the left of $b$, and assume there is no $xy$-monotone path between $a$ and $b$. We let *$x$-first*$(a, b)$ denote a particular shortest path from $a$ to $b$, obeying the following rule, which prioritizes among the motion directions: in traveling from $a$ to $b$ we always prefer to go to the *right*, if possible (i.e., if by moving rightwards from the current location we do not penetrate into an obstacle or lose our ability to obtain a shortest path); if it is not possible to go to the right while remaining on an optimal path, we go to the left, if possible. If both rightwards and leftwards motion are not possible, then we go either upwards (next choice) or downwards (last choice). While $path(a, b)$ is, in general, highly nonunique, *$x$-first*$(a, b)$ is a unique optimal path. We define *$y$-first*$(a, b)$ similarly.

Notice that, in general, the last segment in *$x$-first*$(a, b)$ (resp., *$y$-first*$(a, b)$) will be vertical (resp., horizontal). (This is not true in some special cases, e.g., if $b$ lies on the bottom edge of an obstacle or on an extension of the bottom edge, and if the obstacle is reached via its left edge.)

**Lemma 2.4** *Let $p_1$ and $p_2$ be two points in $\mathcal{P}$, and let $q$ be a point in the free space. Assume $q$ is not blocked in direction $\rho$ by an obstacle, where $\rho$ is one of the 4 directions right, left, up, down. Then by moving $q$ an infinitesimal distance $\varepsilon$ in direction $\rho$, (i) $d(p_1, q)$ and $d(p_2, q)$ each either increases or decreases by $\varepsilon$, and (ii) the difference $d(p_1, q) - d(p_2, q)$ either does not change, or increases or decreases by $2\varepsilon$. (See Figure 1.)*

Let $q \in \mathcal{F}$ and let $\mathcal{P}_{left}$ (resp., $\mathcal{P}_{right}$) denote the subset of $\mathcal{P}$ consisting of the points that lie to the left (resp. right) of $q$, and for which there exists an $x$-monotone path ending at $q$. The *farthest point from $q$ on the left* (resp., *right*) is the point in $\mathcal{P}_{left}$ (resp. $\mathcal{P}_{right}$) obtained

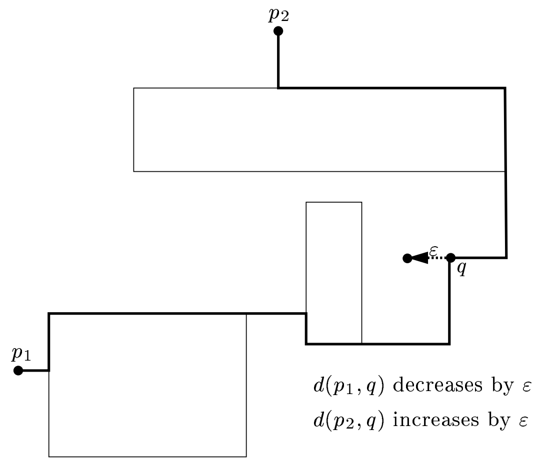


Figure 1: Lemma 2.4: the difference $d(p_1, q) - d(p_2, q)$ decreses by $2\varepsilon$.

as follows. For each of the points in $\mathcal{P}_{left}$ (resp. $\mathcal{P}_{right}$), we construct a shortest $x$-monotone path to $q$. The point with the longest such path is the farthest point from $q$ on the left (resp. right). Similarly, we define the *farthest point from q from below* and the *farthest point from q from above*, using the sets $\mathcal{P}_{below}$ and $\mathcal{P}_{above}$ consisting of the points in $\mathcal{P}$ lying below and above $q$ respectively, and for which there exists a $y$-monotone path ending at $q$.

**Claim 2.5** *The point in $\mathcal{P}$ that is the farthest from $q$ is among the four points that are the farthest from $q$ on the left, right, below, and above.*

**Proof:** Let $p \in \mathcal{P}$ be the farthest point from $q$, and let $path(p, q)$ be any shortest path from $p$ to $q$. By Claim 2.1, $path(p, q)$ is either $x$-monotone or $y$-monotone. Without loss of generality, assume that $path(p, q)$ is $x$-monotone and that $p$ lies to the left of $q$. Then $p \in \mathcal{P}_{left}$ and $p$ is clearly the farthest point from $q$ on the left (applying Corollary 2.3). $\square$

Our first goal (Section 4) is to construct a data structure for answering farthest neighbor queries: Given a query point $q$, find which of the points in $\mathcal{P}$ is the farthest from $q$. The claim above allows us to construct four separate data structures, one for finding the farthest point from $q$ on the left, one for finding the farthest point from $q$ on the right, etc., and to answer a farthest point query by performing a query in each of the four data structures and selecting the farthest of the four candidates. We shall describe the data structure for finding the farthest point on the left; the other cases are analogous.

## 3 Two structure theorems

For a point $a$ in the plane, we draw lines of slope 45 and 135, respectively, passing through $a$. These lines partition the plane into four quarter planes which we call the *right, left, top,* and *bottom regions* of $a$. The following theorem is an immediate corollary of Theorem 6.1, which is stated and proven in Section 6.

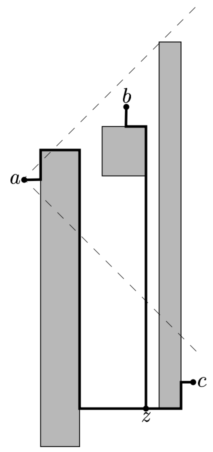


Figure 2: Theorem 3.1: $d(a, c) > d(b, c)$.

**Theorem 3.1** *Let $a$ be a point in the free space, and let $b$ be a point in the free space that lies in the (interior of the) right region of $a$. Let $c$ be a point in the free space that lies to the right of $b$ (not necessarily in the right region of $a$). If there exists an $x$-monotone path from $b$ to $c$, then $d(a, c) > d(b, c)$. (See Figure 2.)*

The above theorem is important from a practical point of view. Let $\mathcal{X}^-$ denote the subset of $\mathcal{P}$ consisting of the points that do not lie in the right region of any other point in $\mathcal{P}$. In other words, if we turn the input scene clockwise by 45 degrees, then the set $\mathcal{X}^-$ consists of all the dominating points. The sets $\mathcal{X}^+$, $\mathcal{Y}^-$, and $\mathcal{Y}^+$ are defined similarly, using the left regions, top regions, and bottom regions, respectively. Recall that we intend to construct a data structure for farthest neighbor queries in $\mathcal{P}$ that is composed of four substructures — for finding the farthest neighbor on the left, on the right, from below and from above. The above theorem allows us, for example, to construct the substructure for finding the farthest neighbor on the left using the subset $\mathcal{X}^-$ rather than $\mathcal{P}$. Since if $p \in \mathcal{P}$ is the farthest neighbor of the query point $q$, then according to Claim 2.5 it is the farthest from $q$ either on the left, on the right, from below, or from above. Assume $p$ is the farthest from $q$ on the left, then $p$ necessarily belongs to $\mathcal{X}^-$. Since, otherwise, there exists another point $p' \in \mathcal{P}$ such that $p$ lies in the right region of $p'$, and, by Theorem 3.1, $d(p', q) > d(p, q)$. (Notice, however, that if $p$ is only the farthest from $q$ on the left in $\mathcal{P}$ but not the farthest from $q$ in $\mathcal{P}$, then it is possible that $p \notin \mathcal{X}^-$.)

This can lead to a significant reduction in the size and construction time of the data structure. For example, in a set of points drawn randomly within the unit square, the expected size of the set $\mathcal{X}^-$ is $O(\sqrt{n})$.

Another consequence is, that if we are searching for the pair of points in $\mathcal{P}$ that defines the *diameter* of $\mathcal{P}$, then we only need to consider pairs of the form $(a, b)$ where $a \in \mathcal{X}^-$ and $b \in \mathcal{X}^+$ or $(c, d)$ where $c \in \mathcal{Y}^-$ and $d \in \mathcal{Y}^+$. Indeed, let $u, v$ be the pair defining the diameter, where $u$ is to the left of $v$, and assume $path(u, v)$ is $x$-monotone. Then, $u \in \mathcal{X}^-$, since otherwise there exists a point that is farther away from $v$, and similarly, $v \in \mathcal{X}^+$.

The sets $\mathcal{X}^-$ and $\mathcal{X}^+$ (alternatively, $\mathcal{Y}^-$ and $\mathcal{Y}^+$) may have some points in common. However, the intersection between $\mathcal{X}^*$ and $\mathcal{Y}^*$, where $* \in \{-, +\}$, consists of a single point, assuming general position (since any such point must lie on a supporting line of slope $\pm 45$

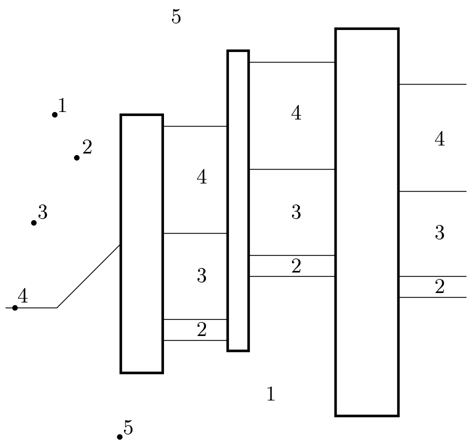


Figure 3: Construction showing that the farthest neighbor Voronoi diagram can have complexity $\Omega(mn)$.

degrees). The (up to) four *extreme* points $\mathcal{X}^- \cap \mathcal{Y}^-$, $\mathcal{X}^- \cap \mathcal{Y}^+$, $\mathcal{X}^+ \cap \mathcal{Y}^-$, and $\mathcal{X}^+ \cap \mathcal{Y}^+$ are the only points that appear in the $L_1$ farthest neighbor Voronoi diagram of $\mathcal{P}$, when ignoring the obstacles. The combinatorial complexity of this diagram is known to be constant. As mentioned already, in our setting the $L_1$ farthest neighbor Voronoi diagram can have combinatorial complexity $\Omega(mn)$, and the number of points with a nonempty region can be $\Omega(n)$.

Our construction of the data structure for finding the farthest neighbor on the left uses the sweep-line technique for a vertical line. Let $l$ be a vertical line. We are interested in the partitioning of $l - \cup\mathcal{R}$ into maximal segments, such that there exists a unique farthest neighbor $p$ on the left for all points in the interior of a segment. That is, for each point $q$ in the interior of a segment, $p$ is the farthest point from $q$, among all points in $\mathcal{X}^-$ that lie to the left of $l$ and can be connected to $q$ by an $x$-monotone path. Notice that in degenerate cases we may have more than one point associated with a single segment. The following theorem is proven in the full version of this paper.

**Theorem 3.2** *Let $l$ be a vertical line, and let $p_1, p_2$ be two points in $\mathcal{X}^-$ to the left of $l$, such that $p_{1,y} > p_{2,y}$. Let $q_1, q_2$ be two points on $l$ such that (i) There exists an $x$-monotone path from $p_i$ to $q_j$, $i, j \in 1, 2$. (ii) $d(p_1, q_1) > d(p_2, q_1)$, i.e., $p_1$ is farther from $q_1$ than $p_2$. (iii) $d(p_2, q_2) > d(p_1, q_2)$, i.e., $p_2$ is farther from $q_2$ than $p_1$. Then $q_{1,y} < q_{2,y}$.*

**Corollary 3.3** *We can generalize the above theorem to any number $k$ of points $p_1, \ldots, p_k$.*

# 4 Farthest neighbor queries

In this section we describe a data structure for "farthest point on the left" queries in $\mathcal{X}^-$. That is, for a query point $q$, find the farthest point from $q$ among all points in $\mathcal{X}^-$ that lie to the left of $q$ and for which there exists an $x$-monotone path connecting them to $q$. As shown above, by constructing this data structure and its three symmetric data structures, we obtain a data structure for farthest point queries in $\mathcal{P}$.

This approach for obtaining a data structure for farthest neighbor queries in $\mathcal{P}$, enables us to avoid the computation of the $L_1$ farthest neighbor Voronoi diagram, which seems more difficult (although probably possible within the same time bound). We only mention at this point that the combinatorial complexity of this diagram is $\Omega(mn)$, as follows from the example depicted in Figure 3.

**Observation 4.1** *It suffices to construct a data structure for farthest point on the left queries in $\mathcal{X}^-$ that is guaranteed to find the correct answer $p$, whenever there is* no *$xy$-monotone path from $p$ to $q$ (while there is an $x$-monotone path). If, in fact, the farthest point on the left, $p$, is one for which there* is *an $xy$-monotone path to $q$, then the distance from $p$ to $q$ cannot exceed the distance from the farthest from $q$ among the four extreme points ($\mathcal{X}^- \cap \mathcal{Y}^-$, $\mathcal{X}^- \cap \mathcal{Y}^+$, $\mathcal{X}^+ \cap \mathcal{Y}^-$, and $\mathcal{X}^+ \cap \mathcal{Y}^+$); these four points can be checked separately.*

Let $l(x)$ denote the vertical line through the point $(x, 0)$. For a point $p = (x_0, y_0) \in \mathcal{X}^-$ and for $x > x_0$, we define the distance function $f_{p,x}$ from $p$ to $l(x)$: Let $q = (x, y)$ be a point on $l(x)$ that does not lie in the interior of an obstacle, then $f_{p,x}(y) = d(p, q)$, if there exists an $x$-monotone path from $p$ to $q$; otherwise, $f_{p,x}(y)$ is not defined (see Figure 4). The *front* of $p$ (at $x$, $x > x_0$) is a set of points obtained as follows. If the horizontal segment from $p$ to $l(x)$ does not enter an obstacle, then the front of $p$ consists of the point $p$ itself. Otherwise, from each minimum point $q(x, y)$ of $f_{p,x}$ we shoot a horizontal ray to the left and add the point that is hit by the ray to the front of $p$.

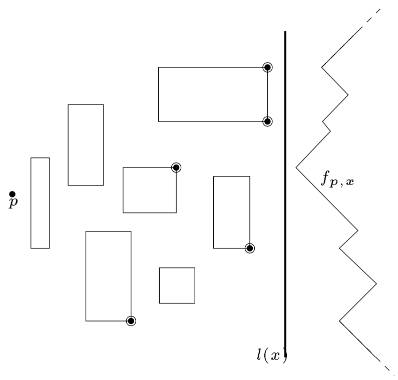


Figure 4: Function $f_{p,x}$ and the front of $p$.

The following claim implies that the front of $p$ is well defined.

**Claim 4.2** *Let $q(x, y)$ be a minimum point of $f_{p,x}$, then $q(x, y)$ lies opposite a right corner of an obstacle to the left of $l(x)$. In other words, if $a$, $a \neq p$, belongs to the front of $p$, then $a$ is a right corner of an obstacle.*

We construct a balanced binary search tree for the front of $p$, where the front points are inserted by their $y$-coordinate. With each front point we store its distance from $p$, and a pointer to the corresponding obstacle.

It is now easy to determine in $O(\log m)$ time the distance between $p$ and any point $q$ on $l(x) - \cup\mathcal{R}$ for which there exists an $x$-monotone path starting at $p$. If $q$ lies above (alternatively, below) all front points, then there exists a shortest path from $p$ to $q$ that passes through the highest (resp., lowest) front point. This shortest path is $xy$-monotone and therefore $d(p,q) = (q_x - p_x) + |q_y - p_y|$. Otherwise, let $a$ and $b$ be the front points immediately above and below $q$, respectively. There exists a shortest path from $p$ to $q$ that passes through either $a$ or $b$, and $d(p,q) = \min\{d(p,a) + (q_x - a_x) + (a_y - q_y), d(p,b) + (q_x - b_x) + (q_y - b_y)\}$. To prove this we first observe that there exists a shortest path from $p$ to $q$ that passes through one of the front points. Next we show that if neither the path through $a$ nor the path through $b$ is a shortest path, then either $a$ or $b$ is not a front point, which is impossible.

Since we are using a sweep-line technique with a vertical line moving from left to right, we must show how to update the front of $p$ when the sweep line $l$ reaches a right edge of an obstacle. (When it reaches the left edge of an obstacle, we do nothing.)

**Updating $p$'s front at the right edge $e$ of an obstacle $R$.** Let $b$ and $t$ be the bottom and top endpoints, respectively, of $e$, and let $a_{min}$ and $a_{max}$ be the lowest and highest points, respectively, in the front of $p$.

1. If $t$ is lower than $a_{min}$ or $b$ is higher than $a_{max}$, that is, if $R$ lies completely below or completely above the front of $p$, then there is no change in the front of $p$.

2. If the edge $e$ lies in between two consecutive front points, that is, if the front point immediately below $b$ is also the one immediately below $t$ and the front point immediately above $t$ is also the one immediately above $b$, then there is no change in the front of $p$.

3. If $b$ is lower than $a_{min}$ (alternatively, $t$ is higher than $a_{max}$), then add $b$ (resp., $t$) to the front of $p$; $d(p,b) = d(p,a_{min}) + d(a_{min},b) = (b_x - p_x) + (p_y - b_y)$ (resp., $d(p,t) = d(p,a_{max}) + d(a_{max},t) = (t_x - p_x) + (t_y - p_y)$).

4. If $b$ lies in between the two consecutive front points $a$ and $a'$, where $a$ is lower than $a'$, then $d(p,b) = \min\{d(p,a)+d(a,b), d(p,a')+d(a',b)\}$. Now, if $d(p,b) = d(p,a')+d(a',b)$, then $b$ is added as a new front point, since it is a minimum point of $f_{p,x}$, otherwise, it is not added to the front of $p$. Similarly, if $t$ lies in between $a$ and $a'$, where $a$ is lower than $a'$, and $d(p,t) = d(p,a) + d(a,t)$, then $t$ is added as a new front point.

5. Remove from the front of $p$ all points that lie in between $b$ and $t$.

The total time that is spent on maintaining the front of $p$ through the entire sweep is $O(m \log m)$, and therefore the total time spent on maintaining the fronts of all points in $\mathcal{X}^-$ is $O(nm \log m)$.

We use the fronts of the points in $\mathcal{X}^-$ to maintain a suitable representation of the function $F_x(y) = \max\{f_{p,x}(y) \mid p \in \mathcal{X}^- \; ; \; p_x \le x\}$. More accurately, we sweep a vertical line $l$ from left to right. When $l$ reaches a right edge $e$ of an obstacle $R$, we compute the representation of $F_x$ restricted to $e$, and store this representation with $R$. At the end of this process we (almost) obtain the desired data structure, as shown below.

The representation of $F_x$ restricted to $e$ is computed as follows. For each point $p \in \mathcal{X}^-$ that lies to the left of $e$, the function $f_{p,x}$, restricted to the edge $e$, is a piecewise-linear function consisting of at most two pieces (see Figure 5). The endpoints of these pieces can be computed easily in $O(\log m)$ time, using the already updated front of $p$. Thus, we can compute the representation of $F_x$ restricted to $e$ in $O(n \log m + n \log n)$ time. The entire process thus takes $O(mn \log(m+n))$ time.

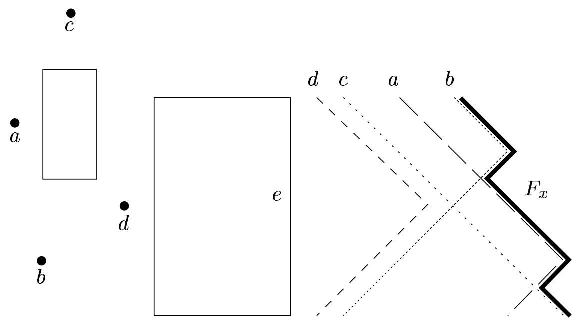


Figure 5: $F_x$ restricted to $e$.

Let $q$ be any point in the free space, for which we want to compute the farthest point on the left in $\mathcal{X}^-$, and let $p \in \mathcal{X}^-$ be the this point. By the observation earlier, we can assume that there is no $xy$-monotone path from $p$ to $q$. Then, when looking leftwards from $q$, one must see an obstacle $R$, such that $p$ lies to the left of $R$ and there is an $x$-monotone path from $p$ to the right edge, $e$, of $R$. It is easy to see that $F_x(q_y) = f_{p,x}(q_y)$, where $x$ is the $x$-coordinate of $e$. Thus we can find $p$ in $O(\log(m+n))$ time by finding $R$ and then searching in the representation of $F_x$ restricted to $e$; $d(p,q) = F_x(q_y) + (q_x - x)$.

**Theorem 4.3** *One can construct a data structure of size $O(mn)$ in $O(mn \log(m+n))$ time, such that the farthest point in $\mathcal{P}$ from a query point $q$ can be found in $O(\log(m+n))$ time.*

If we perform a farthest neighbor query with each of the points in $\mathcal{P}$, then we obtain all farthest neighbors.

**Corollary 4.4** *One can compute all farthest neighbors in $O(mn \log(m+n))$ time.*

**Remark:** Recall that in practice we expect the bounds to be much smaller, since after spending $O(n \log n)$ time in computing the sets $\mathcal{X}^-, \mathcal{X}^+, \mathcal{Y}^-, \mathcal{Y}^+$, $n$ in the above bounds can be replaced by $|\mathcal{X}^-|+|\mathcal{X}^+|+|\mathcal{Y}^-|+|\mathcal{Y}^+|$, which is usually much smaller than $n$, more like $\sqrt{n}$. □

# 5 Computing the center

We begin with a series of observations and claims concerning the location of a center point. Let $c$ be a center point, and let $r$ be the *radius*, i.e., $r = \max\{d(p,c) \mid p \in \mathcal{P}\}$. A point $p \in \mathcal{P}$ for which $d(p,c) = r$ is called an *anchor* point (for $c$). There are at least two anchor points, or, in other words, $c$ lies on an edge of the $L_1$ farthest neighbor Voronoi diagram. Otherwise, if there were only one anchor point $p$, we could move $c$ by an infinitesimal distance towards $p$, thus reducing the radius.

By Theorem 3.1, a point in $\mathcal{P}$ that does not belong to one of the sets $\mathcal{X}^-, \mathcal{X}^+, \mathcal{Y}^-, \mathcal{Y}^+$ cannot be an anchor point. Indeed, let $p$ be such a point, and assume for example that the shortest path from $p$ to $c$ is $x$-monotone and that $p$ lies to the left of $c$. Then, since $p \notin \mathcal{X}^-$, there exists a point $p' \in \mathcal{P}$ such that $p$ is in the right region of $p'$. But according to Theorem 3.1, $d(p',c) > d(p,c)$, so $p$ is not an anchor point.

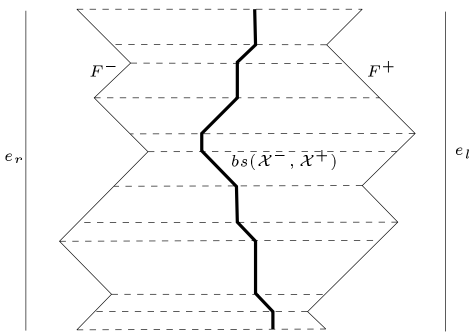


Figure 6: A part of $bs(\mathcal{X}^-, \mathcal{X}^+)$ in which the two relevant $F$-functions are fixed.

Before we further restrict the location of a center point, we introduce the notion of the bisector of $\mathcal{X}^-$ and $\mathcal{X}^+$ (alternatively of $\mathcal{Y}^-$ and $\mathcal{Y}^+$).

### 5.1 Bisectors

The set $bs(\mathcal{X}^-, \mathcal{X}^+)$ consists of all points $q$ in the free space for which the distance between $q$ and its farthest point on the left is equal to the distance between $q$ and its farthest point on the right, where, as before, the farthest point on the left (alternatively, right) is the farthest point among all points that lie to the left (resp., right) of $q$ and for which there exists an $x$-monotone path leading to $q$. The set $bs(\mathcal{Y}^-, \mathcal{Y}^+)$ is defined analogously.

**Claim 5.1** *$bs(\mathcal{X}^-, \mathcal{X}^+)$ is a connected set, broken up into pieces by obstacles, which consists of vertical segments and segments of slope 45 or 135 degrees.*

Figure 6 shows a part of $bs(\mathcal{X}^-, \mathcal{X}^+)$ in which the two relevant $F$-functions ($F^-$ defined on the right edge $e_r$ of some obstacle and $F^+$ defined on the left edge $e_l$ of some obstacle) are fixed. It is clear that the complexity of such a part is $O(n)$, since the complexity of each of the $F$-functions is $O(n)$.

**Claim 5.2** *The complexity of $bs(\mathcal{X}^-, \mathcal{X}^+)$ is $O(mn)$. One can compute $bs(\mathcal{X}^-, \mathcal{X}^+)$ in $O(mn \log(m+n))$ time.*

**Proof:** To compute $bs(\mathcal{X}^-, \mathcal{X}^+)$ we use the data structures $D^-$ and $D^+$ for finding the farthest neighbor on the left and on the right, respectively. When traversing $bs(\mathcal{X}^-, \mathcal{X}^+)$ from top to bottom, the number of pairs of $F$-functions, one from $D^-$ and one from $D^+$, that determine at least one part of the bisector is $O(m)$. Since each pair determines a single part of complexity $O(n)$, or, possibly, several parts of total complexity $O(n)$, the total complexity of $bs(\mathcal{X}^-, \mathcal{X}^+)$ is $O(mn)$.

Now let $y_{top}$ be the top endpoint of the $y$-range of the underlying scene. We first find $x_{top}$, the $x$-coordinate of the point of $bs(\mathcal{X}^-, \mathcal{X}^+)$ with $y$-coordinate $y_{top}$. Since the shortest path from any point $p \in \mathcal{P}$ to any point on the horizontal line through $y_{top}$ is $xy$-monotone, it is enough to consider the four extreme points in order to determine $x_{top}$. After determining $x_{top}$, we compute $bs(\mathcal{X}^-, \mathcal{X}^+)$, segment by segment, beginning from the point $(x_{top}, y_{top})$. In order to compute the next segment whose top endpoint is already known, we need to

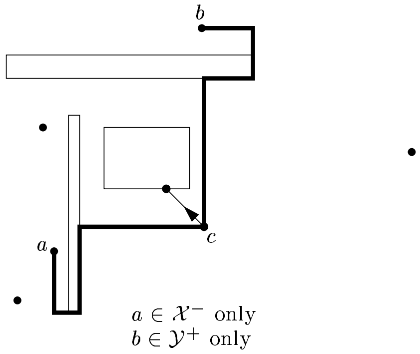


Figure 7: Proof of Theorem 5.4.

consult the appropriate $F_x$ functions, one from $D^-$ and one from $D^+$. The total construction time is dominated by the construction time of the data structures $D^-$ and $D^+$ which is $O((m+n)\log(m+n))$. $\square$

## 5.2 The location of the center point

**Claim 5.3** *If all anchor points of a center point $c$ belong to the set $\mathcal{X}^-$ (alternatively, to one of the sets $\mathcal{X}^+$, $\mathcal{Y}^-$, $\mathcal{Y}^+$) and not to any of the other three sets, then $c$ must lie on the boundary of one of the obstacles.*

**Proof:** By Theorem 3.1 the shortest paths from the anchor points to $c$ cannot be $y$-monotone, and the anchor points must be to the left of $c$. Consider the *$y$-first* paths from the anchor points to $c$. If $c$ is not on a right edge of an obstacle, then the last segment in these paths is horizontal and directed rightwards towards $c$. Thus, by picking a point $c'$ slightly to the left of $c$, we can reduce the radius, which is of course impossible. We conclude that $c$ must lie on a right edge of an obstacle. $\square$

**Theorem 5.4** *There exists a center point that either lies on the boundary of an obstacle, or on one of the two bisectors $bs(\mathcal{X}^-, \mathcal{X}^+)$, $bs(\mathcal{Y}^-, \mathcal{Y}^+)$.*

**Proof:** Let $c$ be a center point not on the boundary of an obstacle and not on one of the bisectors. Then its anchor points belong to two adjacent sets, e.g., $\mathcal{X}^-$ and $\mathcal{Y}^+$ (see Figure 7). We show that it is possible to move $c$ along a segment of slope 135, without increasing the distance to the farthest point in $\mathcal{P}$, until one of the above conditions is satisfied. $\square$

The theorem above allows us to restrict our search for a center point to the two bisectors $bs(\mathcal{X}^-, \mathcal{X}^+)$ and $bs(\mathcal{Y}^-, \mathcal{Y}^+)$ and to the boundaries of the obstacles. (Actually, only the obstacles that break up the bisectors into connected pieces need to be checked).

**Theorem 5.5** *One can find a center point in $O(mn\log(m+n))$ time.*

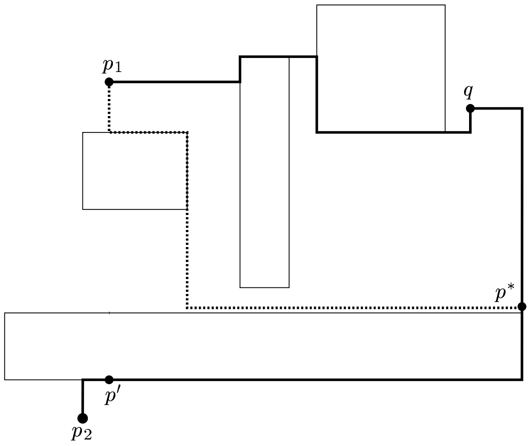


Figure 8: Proof of Theorem 6.1.

**Proof:** Consider, e.g., a vertical segment $s$ of $bs(\mathcal{X}^-, \mathcal{X}^+)$. When moving along $s$ from top to bottom, the farthest neighbors $p_l, p_r$ on the left and on the right, respectively, do not change, and the distance to $p_l$ is equal to the distance to $p_r$. Let $p_a$ be the farthest point from above from the top endpoint of $s$. Then $p_a$ does not change when moving down $s$. Similarly, $p_b$, the farthest point from below from the bottom endpoint of $s$, does not change when moving up $s$. Therefore, in order to determine the best location on $s$, it is enough to consider these four points. $\square$

**Remark:** It is possible to compute all center points (i.e., the center set) within the same time bound. The center set consists of a union of $\Theta(mn)$ line segments (each forming an angle of $\pm 45$ degrees with the horizontal) [5], and one can show that each of these segments has an endpoint that is either on one of the bisectors or on the boundary of an obstacle. Thus, for each center point that is found during the search above, we need to find its corresponding segments. This can be done within the same time bound. $\square$

## 6 Approximation results

We first prove a theorem that is the basis for the subsequent approximations; Theorem 3.1 is a corollary of this theorem.

**Theorem 6.1** *Let $p_1, q$ be two points, $p_1$ to the left of $q$, such that, there exists an $x$-monotone path from $p_1$ to $q$. Let $p_2$ be any point not to the right of $p_1$. Then*

$$d(p_2, q) \geq d(p_1, q) + (p_{1,x} - p_{2,x}) - |p_{1,y} - p_{2,y}| \ .$$

*Thus, if $|p_{1,y} - p_{2,y}| \leq \varepsilon d(p_1, q)$, then $d(p_2, q) \geq (1 - \varepsilon) d(p_1, q)$.*

**Proof:** Let $l$ be the vertical line through $p_1$. Let $path(p_2, q)$ denote a shortest path from $p_2$ to $q$, and let $p'$ be the first intersection point of $path(p_2, q)$ with $l$; see Figure 8. We first prove that

$$d(p', q) \geq d(p_1, q) - |p_{1,y} - p'_y| \ . \tag{1}$$

Assume, e.g., that $p'$ is below $p_1$. Let $path(p', q)$ be an arbitrary shortest path from $p'$ to $q$. We shall find (below) a point $p^*$ on $path(p', q)$, such that there exists an $xy$-monotone path from $p_1$ to $p^*$. Given such a point we have

$$\begin{aligned} d(p', q) &= d(p', p^*) + d(p^*, q) \\ &\geq |p'_x - p^*_x| + |p'_y - p^*_y| + d(p^*, q) \\ &= |p_{1,x} - p^*_x| + |p'_y - p^*_y| + d(p^*, q) \ . \end{aligned}$$

Since $|p'_y - p^*_y| \geq |p_{1,y} - p^*_y| - |p_{1,y} - p'_y|$, we have

$$\begin{aligned} d(p', q) &\geq |p_{1,x} - p^*_x| + |p_{1,y} - p^*_y| \\ &\quad - |p_{1,y} - p'_y| + d(p^*, q) \\ &= d(p_1, p^*) + d(p^*, q) - |p_{1,y} - p'_y| \\ &\geq d(p_1, q) - |p_{1,y} - p'_y| \ . \end{aligned}$$

It remains to show that $p^*$ exists. If $path(p', q)$ is $x$-monotone, then we simply move down from $p_1$ bypassing obstacles from the right until we reach $path(p', q)$. If $path(p', q)$ is $y$-monotone, we need to distinguish among several cases. If $q$ is below $p_1$, we proceed as in the $x$-monotone case. Otherwise, if $p_1$ lies to the left of $path(p', q)$, we move right bypassing obstacles from below, and if $p_1$ lies to the right of $path(p', q)$, we move left bypassing obstacles from below.

We now use Equation 1 to complete the proof.

$$\begin{aligned} d(p_2, q) &= d(p_2, p') + d(p', q) \\ &\geq d(p_2, p') + d(p_1, q) - |p_{1,y} - p'_y| \\ &\geq |p_{2,x} - p'_x| + |p_{2,y} - p'_y| \\ &\quad + d(p_1, q) - |p_{1,y} - p'_y| \ . \end{aligned}$$

But $|p_{2,y} - p'_y| - |p_{1,y} - p'_y| \geq -|p_{1,y} - p_{2,y}|$, so

$$\begin{aligned} d(p_2, q) &\geq |p_{2,x} - p'_x| + d(p_1, q) - |p_{1,y} - p_{2,y}| \\ &\geq d(p_1, q) + (p_{1,x} - p_{2,x}) - |p_{1,y} - p_{2,y}|. \end{aligned}$$

□

## 6.1 Approximating the diameter

Assume the diameter $D$ of the input scene is determined by the points $s, t \in \mathcal{P}$, where $s$ lies to the left of $t$, and assume there exists an $x$-monotone path from $s$ to $t$ (implying that all shortest paths between $s$ and $t$ are $x$-monotone, see Section 2). For any fixed $\varepsilon > 0$, we show how to find a pair of points $a, b \in \mathcal{P}$, such that $d(a, b) \geq (1 - \varepsilon)D$.

The following claim is straightforward.

**Claim 6.2** *Let $p$ be any point in $\mathcal{P}$, and let $q$ be the farthest point from $p$ among the points in $\mathcal{P}$. Then $\frac{D}{2} \leq d(p,q) \leq D$.*

We apply a very simple algorithm, as follows. We divide the $y$-range of the input scene into $O(1/\varepsilon)$ horizontal slabs, each of height $h$, where $\varepsilon D/2 \leq h \leq \varepsilon D$. In each of these slabs, we select the leftmost point, $a \in \mathcal{P}$, and compute its farthest neighbor. (This can be done in time $O((m+n)\log m)$, see Section 4.) Finally, we report the maximum among the $O(1/\varepsilon)$ farthest-neighbor distances. We claim that this results in an approximately maximum distance among pairs of points for which there is an $x$-monotone shortest path. We perform a similar calculation, using vertical slabs, to optimize (approximately) over pairs of points joined by a $y$-monotone shortest path.

Notice that by Claim 6.2, if we take any point $p \in \mathcal{P}$, and find the farthest point $q$ from $p$ among the points in $\mathcal{P}$, then $\varepsilon d(p,q) \in [\varepsilon D/2, \varepsilon D]$. Thus the value $h$ can be found in time $O((m+n)\log m)$ as well.

Consider the horizontal slab that contains the point $s$ (the left endpoint of the diameter pair). Let $a$ be the leftmost point in this slab, and let $b \in \mathcal{P}$ be the farthest point from $a$. Then we claim

**Claim 6.3** $d(a,b) \geq (1-\varepsilon)(s,t)$ .

**Proof:** Since $b$ is the farthest point in $\mathcal{P}$ from $a$, we have $d(a,b) \geq d(a,t)$. Also, $|s_y - a_y| \leq \varepsilon D = \varepsilon d(s,t)$. From Lemma 6.1, $d(a,t) \geq (1-\varepsilon)d(s,t)$. Combining the two inequalities we obtain that $d(a,b) \geq (1-\varepsilon)d(s,t)$. □

**Theorem 6.4** *Given $\varepsilon$, $\varepsilon > 0$, one can find a pair of points $a, b \in \mathcal{P}$, such that $d(a,b) \geq (1-\varepsilon)D$, where $D$ is the diameter of $\mathcal{P}$, in $O(n\log n + \frac{1}{\varepsilon}(n+m)\log m)$ time.*

## 6.2 Finding an approximate center point

Let $c$ be a center point, and let $r$ be the "radius", i.e., $r = \max\{d(p,c) \mid p \in \mathcal{P}\}$. Notice that $D/2 \leq r \leq D$. For any fixed $\varepsilon > 0$, we show how to find a point $c'$, such that $\max\{d(p,c') \mid p \in \mathcal{P}\} \leq (1+\varepsilon)r$.

Divide the $y$-range of the input scene into $O(1/\varepsilon)$ horizontal slabs, each of height $h$, where $\varepsilon D/8 \leq h \leq \varepsilon D/4$. In each of these slabs, select the leftmost and rightmost points. Next, divide the $x$-range into $O(1/\varepsilon)$ vertical slabs, each of width $h$, and select in each of them the topmost and bottommost points. Let $\mathcal{P}'$ be the resulting set of $O(1/\varepsilon)$ selected points. Compute a center point $c'$ for $\mathcal{P}'$ (in time $O((m/\varepsilon)\log(m+1/\varepsilon))$, by our earlier results). Let $r'$ be the radius of $\mathcal{P}'$; clearly $r' \leq r$. Finally, compute the distance between $c'$ and each of the points in $\mathcal{P}$. Let $p \in \mathcal{P}$ be the point farthest from $c'$. Then

**Claim 6.5** $r \leq d(p,c') \leq (1+\varepsilon)r$ .

**Proof:** The left inequality is immediate. We prove the right inequality. Assume there exists an $x$-monotone path between $p$ and $c'$, and let $S$ be the horizontal slab containing $p$. Assuming, e.g., that $p$ is to the left of $c'$, let $p_l$ be the leftmost point in $S$. Notice that we may apply Theorem 6.1 to the triple $p, c', p_l$, since we have that

$$|p_y - p_{l,y}| \leq \varepsilon D/4 \leq \frac{\varepsilon}{2} r \leq \frac{\varepsilon}{2} d(p,c') .$$

Thus, applying Theorem 6.1, we obtain

$$d(p,c') \leq \frac{d(p_l,c')}{1-\frac{\varepsilon}{2}} \leq \frac{r'}{1-\frac{\varepsilon}{2}} \leq \frac{r}{1-\frac{\varepsilon}{2}} \leq (1+\varepsilon)r \ .$$

□

**Theorem 6.6** *Given $\varepsilon$, $\varepsilon > 0$, one can find in $O(n \log(m+n) + (m/\varepsilon)\log(m+1/\varepsilon))$ time a point $c'$, such that $d(p,c') \leq (1+\varepsilon)r$, for any point $p \in \mathcal{P}$, where $r$ is the radius of $\mathcal{P}$.*

# 7 Acknowledgments

We thank Horst Hamacher and Kathrin Klamroth for their valuable input on prior work in facility location problems in the presence of barriers.

# References

[1] B. Aronov, M. van Kreveld, R. van Oostrum, and K. Varadarajan. Facility location on terrains. In *Proc. 9th Annu. Internat. Sympos. Algorithms Comput.*, volume 1533 of *Lecture Notes Comput. Sci.*, pages 357–366. Springer-Verlag, 1998.

[2] M. Atallah and D. Chen. On parallel rectilinear obstacle-avoiding paths. *Comput. Geom. Theory Appl.*, 3:307–313, 1993.

[3] M. J. Atallah and D. Z. Chen. Parallel rectilinear shortest paths with rectangular obstacles. *Comput. Geom. Theory Appl.*, 1(2):79–113, 1991.

[4] J. Choi and C.-K. Yap. Monotonicity of rectilinear geodesics in $d$-space. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.*, pages 339–348, 1996.

[5] J. S. Choi, C. S. Shin, and S. K. Kim. Computing weighted rectilinear median and center set. In *Proc. 9th Annu. Internat. Sympos. Algorithms Comput.*, volume 1533 of *Lecture Notes Comput. Sci.*, pages 29–38. Springer-Verlag, 1998.

[6] K. L. Clarkson, S. Kapoor, and P. M. Vaidya. Rectilinear shortest paths through polygonal obstacles in $O(n(\log n)^2)$ time. In *Proc. 3rd Annu. ACM Sympos. Comput. Geom.*, pages 251–257, 1987.

[7] P. J. de Rezende, D. T. Lee, and Y. F. Wu. Rectilinear shortest paths in the presence of rectangular barriers. *Discrete Comput. Geom.*, 4:41–53, 1989.

[8] P. M. Dearing, H. W. Hamacher, and K. Klamroth. Center problems with barriers and the Manhattan metric. Technical report, No. 667, Dept. of Mathematical Sciences, Clemson University, SC, 1998. submitted to *Naval Research Logistics*.

[9] S. Guha and I. Suzuki. Proximity problems for points on a rectilinear plane with rectangular obstacles. *Algorithmica*, 17:281–307, 1997.

[10] K. Klamroth. Single Facility Location Problems with Barriers. Habilitation thesis, Dept. of Mathematical Sciences, University of Kaiserslautern, Germany, 2000.

[11] J. S. B. Mitchell. An optimal algorithm for shortest rectilinear paths among obstacles. In *Abstracts 1st Canad. Conf. Comput. Geom.*, page 22, 1989.

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

[13] R. Pollack, M. Sharir, and G. Rote. Computing of the geodesic center of a simple polygon. *Discrete Comput. Geom.*, 4:611–626, 1989.

[14] J. Segars, R. *Location Problems with Barriers Using Rectilinear Distance.* PhD thesis, Dept. of Mathematical Sciences, Clemson University, SC, 2000.