Navigating in Unfamiliar Geometric Terrain

(Preliminary Version)

Avrim Blum * Prabhakar Raghavan † Baruch Schieber †

Abstract

Consider a robot that has to travel from a start location s to a target t in an environment with opaque obstacles that lie in its way. The robot always knows its current absolute position and that of the target. It does not, however, know the positions and extents of the obstacles in advance; rather, it finds out about obstacles as it encounters them. We compare the distance walked by the robot in going from s to t to the length of the shortest path between s and t in the scene. We describe and analyze robot strategies that minimize this ratio for different kinds of scenes. In particular, we consider the cases of rectangular obstacles aligned with the axes, rectangular obstacles in more general orientations, and wider classes of convex bodies both in two and three dimensions. We study scenes with non-convex obstacles, which are related to the study of maze-traversal. We also show scenes where randomized algorithms are provably better than deterministic algorithms.

1. Motivation and Results

Practical work on robot motion planning falls into two categories: motion planning through a *known scene*, in which the robot has a complete map of the environment, and motion planning through an *unknown scene* in which an autonomous robot must find its way through a new environment (see, for example, [6, 8, 9, 13, 15] and references therein). Virtually all previous *theoretical* work ([23] and references therein) has focused on the former problem. Papadimitriou and Yannakakis [17] studied the latter problem, which is also the subject of this paper: the design and evaluation of strategies for navigation in an unknown environment. The unfamiliar environment may be either a warehouse or a factory floor whose contents are frequently moved, or a remote terrain such as Mars [21]. The design and evaluation of algorithms for such navigation is a natural algorithmic problem that deserves more theoretical study.

1.1. Model

A *scene* $\mathcal{S}$ consists of a start point s and a target t, together with a set of opaque, impenetrable, non-overlapping obstacles none of which contains s or t. The target t may be a point, or a polygon/polyhedron, or an infinite wall. To avoid certain degeneracies, we assume that a unit circle (unit sphere in three dimensions) can be inscribed in each obstacle; this guarantees that the obstacles have a certain minimum "thickness".

A point robot has to travel from s to t, and it knows both its current absolute position and the position of t. In walking towards t it must circumvent the obstacles in $\mathcal{S}$. The robot does not know the positions and extents of these obstacles in advance; rather, it finds out about obstacles as it encounters them. Where two obstacles touch, we assume that the robot can "squeeze" between them. Thus a scene that consists only of convex obstacles cannot have a non-convex obstacle composed of abutting or overlapping convex obstacles.

The most natural mechanism for the robot to learn about a scene is vision: the robot discovers obstacles as they come into its view, and uses this information to decide how to proceed towards t. For simplicity of exposition in this version, we assume that when the robot first sees an obstacle

*Laboratory for Computer Science, MIT, Cambridge, MA 02139. Supported in part by an NSF graduate fellowship. Part of the work was done while this author was visiting IBM T.J. Watson Research Center. avrim@theory.lcs.mit.edu

†IBM Research Division, T.J. Watson Research Center, Yorktown Heights, NY 10598. {pragh,sbar}@ibm.com

it is given the shape, size and position of the obstacle (even though much of that obstacle may be invisible from where it stands). However, we show how many of our algorithms can be made to work with essentially the same upper bounds (up to a constant factor) under a considerably weaker assumption — a *tactile robot* that learns about obstacles only by bumping into them and moving along them. For this we use variants on the "doubling" strategies of Baeza-Yates *et al.* [1]. Details are omitted in this version.

Let $R(\mathcal{S})$ be the total distance walked by a robot R in going from s to t in scene $\mathcal{S}$, and let $d(\mathcal{S})$ denote the length of the shortest path in the scene between s and t (because of the obstacles, this may be substantially larger than the Euclidean distance between s and t). Let $\mathcal{S}(n)$ denote the set of scenes in which the Euclidean distance between s and t is n. Following the lead of [17], we use as the figure of merit for the robot the *ratio*

$$\rho(R,n) = \max_{\mathcal{S}\in\mathcal{S}(n)} \frac{R(\mathcal{S})}{d(\mathcal{S})},$$

and study its growth as a function of n.

For convenience, we put the scene in Cartesian coordinates, using "north"/"south" to denote the direction of increasing/decreasing y value, "east"/"west" for the direction of increasing/decreasing x value, and "up"/"down" for the direction of increasing/decreasing z value, respectively. The start point s is always assumed to be at the origin.

1.2. Summary of Results

We first consider scenes where t is a point and the obstacles are rectangles with sides parallel to the axes (rather than squares as in [17]). Surprisingly, even this problem turns out to be quite complicated. Any robot purporting to achieve a ratio ρ on such scenes must achieve a ratio of at most ρ on both the following special cases.

1. Scenes in which t is an *infinite line* and the obstacles are oriented rectangles. We call this case the *wall problem.*

2. Scenes in which the obstacles are oriented rectangles that are confined to lie within a square "room". Here s is a point on a wall of the room and t is a point in the room. We call this the *room problem.* This intriguing special case is of interest in its own right as a model for navigation in a bounded region such as a warehouse.

Section 2 describes an optimal algorithm for the wall problem. The algorithm achieves an upper bound of $O(\sqrt{n})$ on the ratio $\rho(R,n)$, matching the lower bound of [17]. To devise this algorithm we develop a general "sweep" paradigm that is fairly natural: a human lost in a strange city would probably do a similar search. We extend this approach to three dimensions, proving a tight bound of $\Theta(n^{2/3})$ for the ratio in this case.

Section 3 considers the room problem. The algorithm for this problem achieves a ratio $\rho(R,n) = 2^{c\sqrt{\log n}}$, for some constant c. We conjecture that this is not optimal, and suspect that a constant ratio is achievable. The approach taken by this algorithm is different from the one taken for the wall problem. Here, we develop a "calliper" method that pins the target down to lie within a sequence of advancing paths. Intriguingly, in the room problem $d(\mathcal{S}) \leq 2n$ for all $\mathcal{S}$. To see this, suppose that s is the south-west corner of the room. Observe that the length of the *greedy* path from the target t to the start point s (a path that travels due south if possible and otherwise due west) is the L_1 distance between s and t (Fig 2). In contrast, going from s to t seems considerably harder, with greedy paths from s not guaranteed to go anywhere near t. Thus getting out of a room is easy, but getting in towards a small target seems to be hard. We extend the room problem to scenes containing arbitrary polygonal obstacles. We show that $d(\mathcal{S})$ can now be as large as $n^{3/2}$. Unlike the case of oriented rectangles the greedy path is no longer guaranteed to find an inexpensive way out of the room. For these scenes we give a lower bound of $\Omega(\sqrt{n})$ on $\rho(R,n)$, for any deterministic algorithm and a randomized algorithm that achieves $\rho(R,n) = O(\sqrt{n})$.

Section 4 shows how to combine our solutions for the wall and room problems to obtain a tight bound of $\Theta(\sqrt{n})$ for point-to-point navigation in scenes consisting of oriented rectangular obstacles. The solution extends to three dimensions, yielding

a tight bound of $\Theta(n^{2/3})$ for navigating through oriented cuboids.

In Section 5 we give a randomized algorithm for certain cases of the wall problem in two and three dimensions. We show that the (expected) ratio of our algorithm is $\leq 2^{c\sqrt{\log n \log\log n}}$ (for some constant c), which is much smaller than the corresponding deterministic lower bound. This demonstrates the power of randomization in navigation.

Section 6 deals with non-convex obstacles (and therefore mazes). We give a lower bound for randomized algorithms, and show that a deterministic algorithm of Rao *et al.* [19] meets this bound. The algorithm is memory-intensive, and so we offer an alternative algorithm that is very simple, memoryless, randomized and achieves the same upper bound in the plane.

We conclude with open problems in Section 7.

1.3. Related Previous Theoretical Work

The first paper to consider the ratio $\rho(R,n)$ is that Papadimitriou and Yannakakis [17]. They proved that when s and t are points in the plane, and all obstacles are squares, $\rho(R,n)$ is at least 1.5, and complement this with an algorithm attaining $\rho(R,n) \leq 1.5 + o(1)$ for all n. They also show that when t is an infinite wall at distance n from s and the obstacles are oriented rectangles, then $\rho(R,n)$ is $\Omega(\sqrt{n})$. Coffman and Gilbert [7] study the performance of simple heuristics in the presence of randomly placed obstacles. Kalyanasundaram and Pruhs [10] consider scenes in which all obstacles have bounded aspect ratios. Lumelsky and Stepanov [13] earlier gave a simple navigation algorithm that guarantees $R(\mathcal{S})$ to be bounded by $d(\mathcal{S})$ plus the sum of the perimeters of all obstacles, with no restrictions on the aspect ratios or the convexity of the obstacles. Their algorithm does not minimize the ratio ρ. Several papers (see [16, 19, 20] and references therein) give algorithms for building up a map of a scene by exploring it entirely. Maze-traversal has received considerable attention in the past in various papers [4, 12, 18], none of which considers the ratio metric.

The ratio measure $\rho(R,n)$ has close connections to the *competitiveness* measure used in the study of on-line algorithms [5, 14, 22]; indeed, our problem resembles an on-line setting in which the obstacles encountered by the robot form a sequence of "requests", and we compare its total cost $R(\mathcal{S})$ to the "off-line cost" $d(\mathcal{S})$. It is therefore worth pointing out some key differences between the models: (a) In the navigation problem the robot has a definite target towards which it moves, while there is no such notion in on-line paging [22], for example. (b) The robot can move back and forth through the scene, re-visiting previously seen obstacles, thus having some control on the requests it encounters in the future. (c) Competitive analysis deals with request sequences of arbitrary (possibly infinite) length, whereas here we have a fixed number of obstacles in the scene. Thus we cannot cast our navigation problem in a standard on-line framework such as the server problem [14] or metrical task systems [5]. Nevertheless, the analogy with on-line algorithms proves useful in the study of randomized navigation (Section 5).

2. The Wall Problem

In this section we consider scenes in which t is an infinite wall at distance n from s, and the obstacles are rectangles whose sides are parallel to the axes. Without loss of generality assume that t is a vertical wall to the east of s. We call the *width* of an obstacle its dimension parallel to the x axis, and the *height* of an obstacle its dimension parallel to the y axis. To avoid degeneracies we assume that the width and the height of every obstacle is at least one.

We present an algorithm that achieves ratio $\rho(R,n) = \Theta(\min\{\sqrt{n}, 1 + \bar{h}/d(\mathcal{S})\})$, where $\bar{h}$ is the *total height* of the obstacles seen by the robot. This matches the lower bound proven in [17], so our algorithm is optimal up to constant factors.

The algorithm maintains four variables: the *window size*, W, a *threshold*, τ, a *sweep direction*, and a *sweep counter*. Initially, W is set to n, the sweep direction is south, and the sweep counter is set to zero. The threshold τ is always set to $W/\sqrt{n}$.

We begin with a high-level view of the algorithm and its analysis. The algorithm maintains a window of varying size around the x-axis. The robot makes $\sqrt{n}$ sweeps in directions alternating between north and south for each window size. Upon com-

pletion of these $\sqrt{n}$ sweeps the window size is doubled. Given a window of size W (which ranges from $y = +W/2$ to $y = -W/2$), the distance walked by the robot in sweeping is $O(W\sqrt{n})$. We show that the shortest path that cuts through all the $\sqrt{n}$ sweeps has length $\Omega(\sqrt{n}\tau) = \Omega(W)$. Let W_f be the window size at the time the robot reaches t. We prove that the total distance walked by the robot is $O(\min\{n + \bar{h}, W_f\sqrt{n}\})$, while $d(\mathcal{S}) = \Omega(W_f)$.

We now describe the algorithm. Starting from point s, the robot travels due east until it either reaches t or hits an obstacle, say at (x, y). Let y_n and y_s be the *absolute values* of the y-coordinates of the north and south corners of the obstacle. Below, we assume that the current sweep direction is south, the other case is symmetric. The next steps are determined by the following rules:

Rule 1: If the distance to the nearest corner is less than τ, then the robot goes "around" the obstacle. Specifically, it travels either south or north to the nearest corner, then east along the width of the obstacle until it reaches the opposite corner (or until it reaches t and stops). Then, it travels along the height of the obstacle arriving at point $(x + w, y)$, where w is the width of the obstacle. (See Fig. 1(a).) From this point it continues to travel due east until it hits the next obstacle.

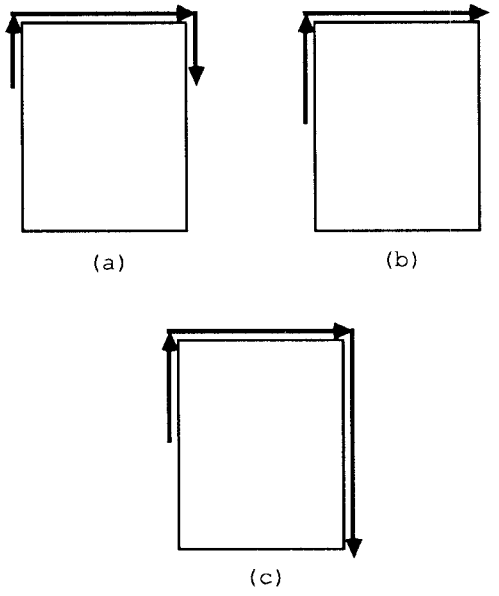

Figure 1: Going around an obstacle

Rule 2: If $y_n > W/2$ and $y_s > W/2$ (i.e., the obstacle extends past both sides of the window), then the robot travels to the corner that extends less past the window side; that is, it goes north if $y_n \leq y_s$ and south otherwise. From this corner it continues to travel east, after setting the sweep counter to one, the window size W to $2(\min\{y_n, y_s\})$, and the threshold τ to $W/\sqrt{n}$. The sweep direction is set to north if the robot is at the south-east corner and to south otherwise. (See Fig. 1(b).)

Rule 3: Otherwise (i.e., the distance to the nearest corner is more than τ, and the obstacle does not extend past both sides of the window), then the robot travels to the nearest corner, and then east along the width of the obstacle until it reaches the opposite corner (or until it reaches t and stops). If the robot is at the south east corner then it continues to travel east. (Recall that the sweep direction is south.) Otherwise, i.e., it is at the north-east corner, it travels south along the height of the obstacle until it reaches either the south east corner of the obstacle or the point $(x + w, -W/2)$, and then proceeds due east. (See Fig. 1(c).) In the latter case it increments the sweep counter by one, and flips the sweep direction. If the counter is greater than $\sqrt{n}$, it resets the counter to zero and doubles the window size and the threshold.

Theorem 1: *The total distance walked by the robot is* $O(\min\{n + \bar{h}, W_f\sqrt{n}\})$.

Proof: Since the robot walks only along the borders of the obstacles and never backtracks, it is clear that the total distance is no more than $O(n + \bar{h})$.

To prove that the distance is also bounded by $O(W_f\sqrt{n})$, we divide the path taken by the robot into three components: (1) horizontal segments, (2) segments walked south and north "along" obstacles using Rule 1, and (3) segments walked south and north using the other rules.

Notice that (i) the total distance walked east is $n = O(W_f\sqrt{n})$, and (ii) since the width of each obstacle is at least one unit, the total distance walked south and north using Rule 1 is bounded by $n\tau_f = O(W_f\sqrt{n})$, where τ_f is the final threshold. It suffices to bound the third component as well. Fix a window size W. The distance walked by the robot using Rules 2,3 to complete one sweep is $O(W)$. Since the robot does at most $\sqrt{n}$ sweeps

for each window size we get that the total distance for a fixed window size is $O(W\sqrt{n})$. The window size is at least doubled each time it is changed, and thus the total distance travelled over all window sizes is also $O(W_f\sqrt{n})$. □

Theorem 2: *The shortest path from s to t, $d(\mathcal{S})$, is $\Omega(W_f)$.*

Proof: Since we are interested in the value of the shortest path up to a constant factor, we may assume that the path consists only of horizontal and vertical segments. The length of the horizontal segments is clearly $\Omega(n)$. If $W_f = n$ then we are done. Assume not; in this case we prove that the length of the vertical segments is $\Omega(W_f)$.

Suppose that the robot has completed at least $\sqrt{n}$ sweeps for some window size. Let W be the largest such window size. We claim that the vertical component of $d(\mathcal{S})$ is $\Omega(W)$. To see this, note that to cut through all the $\sqrt{n}$ sweeps the shortest path may either go "around" all the sweeps walking distance W, or walk a vertical distance of at least τ in each sweep, totaling $\sqrt{n}\tau = W$. If $W_f \leq 2W$ then we are done. Otherwise (i.e., $W_f > 2W$, or there is no window for which the robot has completed $\sqrt{n}$ sweeps), then it must be the case that W_f is determined by rule 2, in which case the shortest path is clearly $\Omega(W_f)$. □

Extension to Three Dimensions. We now very briefly sketch extensions of the sweep paradigm to three dimensions. By an argument similar to that in [17], it can be shown that the ratio for point-to-wall navigation in three dimensions is $\Omega(n^{2/3})$. (Details omitted here.) Our algorithm for point-to-wall navigation matches the lower bound to within a constant factor provided every obstacle is a general *cylinder*: its intersection with every plane perpendicular to the x-axis is either a fixed closed curve (possibly non-convex), or empty. As in the two dimensional case, we maintain a threshold and a window that grow together. The three-dimensional analog of the sweep is a spiral. The spacing of successive orbits of the spiral is chosen to to keep the cost of the robot to within $n^{2/3}$ of the shortest path.

3. The Room Problem

In this section we consider scenes in which the obstacles are oriented rectangles confined to lie within a square room such that no obstacle touches the room walls. The point s is on the border of the room and t is in the center. (See Fig. 2.) Results pertaining to t elsewhere in the room are similar and omitted here. Later, we extend our results to rectangular rooms. Since travel along the room walls is "cheap", we may assume s is in the southwest corner of the room, and for convenience we let t have coordinates (n, n), so the distance from s to t is in fact $n\sqrt{2}$. Also, for simplicity, we assume all obstacles have corners at integral (x, y) points.

Define a *greedy* $\langle +x, +y \rangle$ path to be a path that travels due east if possible and otherwise due north. Similarly, define greedy $\langle +y, +x \rangle$ paths, $\langle +x, -y \rangle$ paths, and so forth, to be ones that travel in the first direction if possible and otherwise the second direction. A *brute-force* $\langle +x \rangle$ path is one that travels due east, going around obstacles in its way along the shorter direction, but otherwise maintaining a constant y coordinate. A *monotone* path from (x_1, y_1) to (x_2, y_2) is one where the x and y coordinates never decrease along the direction of the straight-line path. For example, if $x_2 > x_1$ and $y_2 < y_1$, then the x coordinate will never decrease and the y coordinate will never increase. Notice that a greedy path is always monotone.

We now describe an algorithm achieving $R(\mathcal{S}) = O(n^{3/2})$, and thus $\rho(R, n) = O(\sqrt{n})$. An improvement that uses this algorithm recursively achieves $\rho(R, n) = O(2^{c\sqrt{\log n}})$.

The algorithm maintains the following invariant at the start of each iteration: the robot knows of a monotone path from a point (x_0, n) to a point (n, y_0), where $0 \leq x_0, y_0 \leq n$, that is crossed by no obstacle. Furthermore, it is positioned on a point of this monotone path. We begin with $x_0 = y_0 = 0$, where the known path is just along the room borders. Each loop through the algorithm will increase either x_0 or y_0 by at least an amount $\sqrt{n}$, walking a distance of only $O(n)$. Since each of x_0 and y_0 can be increased by this amount only $\sqrt{n}$ times, the total distance walked by the robot to reach t is $O(n^{3/2})$.

The algorithm consists of four steps. For this

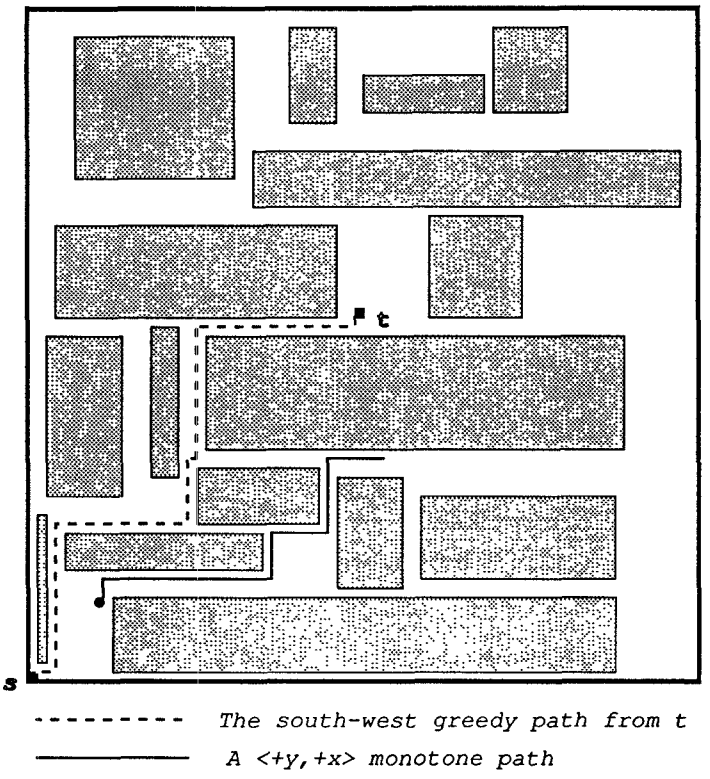

Figure 2: The Room Problem

first version of the algorithm, let $m = \sqrt{n}$.

Algorithm **Oriented-Room-Find**

Step 1: Travel to location $t' = (x_0 + m, y_0 + m)$. If this point happens to be inside an obstacle, or "below" our current monotone path, then any other point to its north-east that is to the south-west of t will do. In particular, if t' is inside an obstacle, then either the north-west or the south-east corner of the obstacle is to the south-west of t and may be used. If t' is below the monotone path, any point on the monotone path (that is not inside an obstacle) to the south-west of t may be used.

Step 2: Make a greedy $\langle +x, +y \rangle$ path from t' until either the x or y coordinate equals n. Without loss of generality, assume that the robot has traveled to the west of t, so the current coordinates are $(\hat{x}, n)$ for $\hat{x} < n$. Notice that $\hat{x} \geq x_0 + m$ since the path was greedy.

Step 3: Let $x_0 \leftarrow \hat{x}$. Travel a greedy $\langle +x, -y \rangle$ path from $(\hat{x}, n)$ until either a point $(n, \hat{y})$ is reached (in which case $y_0 \leftarrow \hat{y}$), or the previous monotone path is hit. In either case we now have a new monotone path from (x_0, n) to (n, y_0) with x_0 increased by at least m. (Similarly, the robot would travel a greedy $\langle -x, +y \rangle$ path if it had hit to the south of target t in step 2.)

Step 4: If $x_0 < n - m$ or $y_0 < n - m$ then go back to Step 1. Otherwise, perform a modified version of Step 1 with $t' = t$, and stop.

The distance walked performing Step 1 is $O(m^2) = O(n)$ since the robot may travel along the monotone boundary to a point with y-coordinate equal to that of t', and then travel in a brute-force $\langle +x \rangle$ path to t'. The brute-force $\langle +x \rangle$ path travels at most $2m$ distance for every unit of progress in the $+x$ direction made, since the lower end of any obstacle encountered cannot extend below the monotone path. Steps 2 and 3 together require travel at most a distance $3n$ since each consists of a single greedy path, and in the two paths together at least one coordinate is non-decreasing. Since these steps are iterated at most $2m$ times, the total distance traveled is at most $O(n^{3/2})$.

We can improve the result to $O(n \cdot 2^{c\sqrt{\log n}})$ by us-

ing algorithm Oriented-Room-Find recursively to reach temporary destination t' in Step 1, and then optimizing the value of m – the x (and y) coordinate distance from the start point to t'. Define $T(n)$ to be the total distance traveled to reach t at x (and y) coordinate distance n from the start point. For a fixed value of m, the distance traveled at each iteration of Step 1 is at most $T(m)$, while the distance traveled at each iteration of Steps 2 and 3 is at most $3n$. The number of iterations is at most $2n/m$, so the recursion is:

$$T(n) \leq \frac{2n}{m}[T(m) + 3n],$$

which solves to $T(n) = n^{1+c/\sqrt{\log n}}$.

3.1. Arbitrary Rectangular Obstacles

What if rectangular obstacles with sides not parallel to the axes are allowed? We begin by proving two theorems that demonstrate the difference between scenes containing only oriented rectangular obstacles, and scenes containing arbitrary rectangular obstacles.

Theorem 3: *There exist scenes $\mathcal{S}$ containing rectangular obstacles whose sides are at arbitrary angles for which $d(\mathcal{S}) \geq \pi n^{3/2}/27$.*

Thus, the length of the shortest path between s and t is not always bounded above by the L_1 distance as in the oriented case.

Theorem 4: *For any deterministic robot R, there exist scenes $\mathcal{S}$ containing rectangular obstacles whose sides are at arbitrary angles for which $\rho(R, n) = \Omega(\sqrt{n})$.*

Thus, the upper bound for oriented rectangles cannot be achieved in this case.

Proof of Theorem 3: Consider $n/9 + 1$ circles centered at t, with radii $n/3 + 3i$, $i = 0, \ldots n/9$. Inscribe in each a regular $\sqrt{n}$-gon. Rotate all the polygons inscribed in circles of radii $n/3 + 3i$ for even i by an angle $\pi/\sqrt{n}$. Each edge of each polygon can now be replaced by a rectangular obstacle of unit width (in the radial direction) and length very nearly the length of that edge. Now, any obstacle-avoiding path between s and t has to walk a distance of at least $2\pi\sqrt{n}/3$ going from a point on the circle with radius $n/3 + 6i$ to a point on the circle with radius $n/3 + 6i + 6$, for $0 \leq i < n/18$. □

Proof of Theorem 4: Consider the scene described in the proof of Theorem 3. We allow a (deterministic) robot to walk from s to t. We now remove from the scene any obstacle not touched by the robot. Let T be the number of obstacles it touches. It can be shown that the robot pays a cost at least $T\sqrt{n}/4$. A pigeonholing argument shows that $d(\mathcal{S}) \leq 4T$. Following a technique from [17], the argument can be made to work even when the robot uses vision. □

We now turn to upper bounds. Define the angle of a rectangle to be the angle of its longest edge. We first describe a modification of algorithm Oriented-Room-Find to handle not just obstacles of angles of 0 and $\pi/2$, but obstacles angled in the range $[0, \pi/2]$ as well, retaining the same performance. Note that we do *not* allow obstacles angled in the remaining range of $(\pi/2, \pi)$. Then, we describe how this new algorithm can be modified for scenes $\mathcal{S}$ where there is a known excluded range (d_1, d_2) of angles (for example, $d_1 = \pi/5$ and $d_2 = \pi/4$). Let $\tilde{n} = n/\alpha$, where $\alpha = d_2 - d_1$. Our algorithm achieves $R(\mathcal{S}) = O(\tilde{n})$, if $\tilde{n} = \Omega(n^{1+\epsilon})$, for some constant ϵ, and $R(\mathcal{S}) = O(\tilde{n} \cdot 2^{c\sqrt{\log \tilde{n}}})$ otherwise. The length of the shortest path in such scenes is $O(n)$. We remark that for "practical" cases it may be enough to consider scenes where there is a known excluded range, since usually the number of different angles of the obstacles in real-life scenes is small.

Finally, we give a randomized algorithm that achieves $\rho(R, n) = \sqrt{n}$ regardless of the angles of the obstacles.

The idea for the situation where angles are in the range $[0, \pi/2]$ is as follows. First, let us treat the obstacles as infinitesimally thin lines, say along a longest edge (the thickness of obstacles can be handled separately). Since every obstacle has angle in the range $[0, \pi/2]$, we can still perform greedy $\langle +x, +y \rangle$ and $\langle +y, +x \rangle$ paths. The problem is that we can no longer make the greedy $\langle +x, -y \rangle$ path required in Step 3 of Oriented-Room-Find. Instead, we will use binary search to find a point (x', y')

either directly north or east of t' with the following property: a greedy $\langle +x, +y \rangle$ path from (x', y') hits a point $(n, \hat{y})$ to the south of t and a greedy $\langle +y, +x \rangle$ path from (x', y') hits a point $(\hat{x}, n)$ to the west of t. So, consider just the condition that we begin with a (*not* necessarily monotone) known path of length $O(n)$ from the line $y = n$ to the line $x = n$ with least y value y_0 and least x value x_0. Then, we will have produced a new such path with either the least x value or least y value increased by m'.

This strategy can be used for a smaller range (d_1, d_2) of excluded angles by just performing a rotation and a coordinate transformation on the space. Essentially, instead of writing t as $n\vec{x} + n\vec{y}$ for orthogonal unit vectors $\vec{x}$ and $\vec{y}$, we may write t as $(n'\vec{d_1} + n''\vec{d_2})$, where $\vec{d_1}$ and $\vec{d_2}$ are unit vectors in the d_1 and d_2 directions. It is not difficult to see that both n' and n'' are $O(n/\alpha)$, where $\alpha = d_2 - d_1$. Let $\tilde{n} = n/\alpha$. The performance of the previous algorithm after the transformation is $R(\mathcal{S}) = O(\tilde{n})$, if $\tilde{n} = \Omega(n^{1+\epsilon})$, for some constant ϵ, and $R(\mathcal{S}) = O(\tilde{n} \cdot 2^{c\sqrt{\log \tilde{n}}})$ otherwise.

Theorem 5: *There is a deterministic algorithm for the room problem with an excluded range of angles α that achieves a ratio $R(\mathcal{S}) = O(\tilde{n})$, if $\tilde{n} = \Omega(n^{1+\epsilon})$, for some constant ϵ, and $R(\mathcal{S}) = O(\tilde{n} \cdot 2^{c\sqrt{\log \tilde{n}}})$ otherwise. Here $\tilde{n} = n/\alpha$.*

Consider now the general case where the angles of the obstacles may be in any range. A simple pigeonholing argument implies that a constant fraction of the ranges $[i\pi/\sqrt{n}, (i+1)\pi/\sqrt{n}]$, for $1 \leq i < \sqrt{n}$ have the property that the total area of the obstacles angled in this range is no more than $2/\sqrt{n}$ of the total area. Thus the total area of obstacles in such a range is always $O(n^{3/2})$.

Consider a randomized algorithm that first guesses such a range. It then applies the above algorithm assuming that there are no obstacles with angles in this range, on actually encountering any obstacle in this range, it just goes around the obstacle at cost at most the area of the obstacle. Thus the total cost paid by the algorithm on such obstacles is $O(n^{3/2})$. The expected total distance walked by this algorithm is given by the recursion:

$$T(n) \leq \frac{2n}{m}\left[T(m) + 3n^{3/2}\right],$$

whose solution is $T(n) = n^{3/2}$.

Theorem 6: *There is a randomized algorithm achieving a ratio that is $O(\sqrt{n})$ for the room problem provided every obstacle is a rectangle within which a unit circle can be inscribed.*

4. Point-to-point Navigation

We combine the algorithms for the wall and room problems to obtain an algorithm for navigation in scenes where t is a point at (n, n) and the obstacles are oriented rectangles with no bound on their extents.

The robot starts by taking a greedy $\langle +x, +y \rangle$ path from s until it reaches a point s' with either x or y coordinate equal to t. Suppose that s' and t have the same y-coordinate. The robot now uses the sweep algorithm for the wall problem to travel to a point on the vertical wall containing t. Notice that the path taken by the robot in the sweep algorithm is a path from s' to a point (n, y_0) where $|y_0 - n| \leq d(\mathcal{S})$ that never decreases in the x direction. This path can be used to obtain a *monotone* path from (n, y_0) to a point (x_0, n) with $x_0 \geq 0$. Finally the robot invokes the algorithm for the room problem to arrive at t using the monotone path as the room walls.

We analyze the distance walked by the robot. The distance traveled using the algorithm for the wall problem is at most $O(\min\{n + \bar{h}, W_f\sqrt{n}\})$, where W_f is the size of the last window considered. The size of the (rectangular) room then created is at most $n \times W_f$. So, using the algorithm from section 3, the distance walked to reach t is $O(W_f\sqrt{n})$. If $W_f = O(n)$ then we are done since the length of the shortest path from s to t is at least $2n$. Suppose that this is not the case. By Theorem 2, the length of the shortest path from s' to t is $\Omega(W_f)$. Since the length of the shortest path from s' to s is at most $2n$, $d(\mathcal{S}) = \Omega(W_f - 2n) = \Omega(W_f)$. We therefore have the following theorem.

Theorem 7: *For two dimensional scenes $\mathcal{S}$ in which s and t are points and every obstacle is a rectangle whose sides are parallel to the axes our algorithm achieves a ratio of $\rho(R, n) = O(\sqrt{n})$.*

Extension to Three Dimensions. As in two dimensions, we give an upper bound for point-to-point navigation that matches the lower bound to within a constant factor provided every obstacle is a cuboid whose sides are parallel to the axes. Also as in two dimensions, our upper bound for point-to-point navigation comes from combining an algorithm for point-to-wall navigation and another for the room problem.

It suffices to combine the three dimensional sweep algorithm with the two-dimensional room algorithm to obtain a three-dimensional navigation algorithm for which we prove:

Theorem 8: *For three dimensional scenes $\mathcal{S}$ in which s and t are points and every obstacle is a cuboid whose sides are parallel to the axes our algorithm achieves a ratio of $\rho(R,n) = O(n^{2/3})$.*

5. The Power of Randomization

We now consider randomized robots that toss coins as they walk from s to t. The scene $\mathcal{S}$ is fixed in advance by an *oblivious* adversary [2] who knows the randomized algorithm, but not the coin tosses made by the robot during a walk. The cost of robot R on scene $\mathcal{S}$ is now a random variable; we thus define the ratio $\rho(R,n)$ to be $\max_{\mathcal{S}\in\mathcal{S}(n)} E[R(\mathcal{S})]/d(\mathcal{S})$. The main result of this section is a randomized algorithm for the wall problem that achieves a ratio that is $2^{O(\sqrt{\log n \log\log n})}$ provided the obstacles are all vertical line segments of integral x-coordinates and the robot is allowed *vision*. Notice that for this situation, the robot can see the entire "column" of obstacles directly in front of it; that is, if the robot is at a point with x-coordinate in the range $(i-1,i)$ for integer i, it can see all obstacles of x-coordinate i. To keep with our previous conventions on the thickness of obstacles, we could equivalently consider obstacles of width between one and two having their left walls only at even x-coordinates; this would still allow the robot to see an entire "column" at once.

The Papadimitriou-Yannakakis lower bound of $\Omega(\sqrt{n})$ still holds for deterministic algorithms for this restricted class of scenes [17]. So, for such scenes, a randomized algorithm is provably better than a deterministic one. We leave as an open question whether one can achieve similar bounds for the more general wall problem.

The idea for the randomized algorithm is to view the problem as a k-server problem on $(k+1)$ equally-spaced points on a line, and then use as a subroutine known randomized strategies [3] for that server problem. In fact, our problem can be better described as a *metrical task system* of [5], but we will use the language of servers here. In the lower bound direction, a recent result of Karloff *et al.* for the server problem shows that even for the special case of scenes we consider, no randomized algorithm can achieve a constant ratio [11].

Theorem 9: *The randomized algorithm below achieves a ratio that is $2^{O(\sqrt{\log n \log\log n})}$ for the wall problem in the plane where the robot uses vision and the obstacles are vertical line segments at integral x-coordinates.*

Proof: We map the navigation problem to a k-server problem on $(k+1)$ equally-spaced points on the line as follows. Let $n = k+1$ and define the spacing between adjacent points on the line to be W/n, where W is the size of a window of y-coordinates considered by the robot. Each point in the server problem corresponds to a range of W/n y-coordinates for the navigation problem. The "hole" (the point without a server) represents the range of y-coordinates currently inhabited by the robot.

We begin with $W = n$ and start the hole at the center of the line. Each time the robot sees a column of obstacles, the robot notes all points in the server problem corresponding to ranges that are completely blocked by obstacles. It then makes enough requests to the server problem on those points so that for any *reasonable* (lazy) algorithm [14] the hole no longer resides on such points. The robot now moves in the y direction to the range corresponding to the new location of the hole. It then moves an additional at most $W/n+1$ distance to find a point in that range not blocked by any obstacle, and from there goes forward in the $+x$ direction to the next column. So, the distance moved by the robot is at most the on-line server cost, plus $W/n+1$ for each unit moved in the $+x$ direction.

There is a randomized strategy for k servers on $k+1$ equally-spaced points on the line that achieves a competitiveness at most $2^{c\sqrt{\log k \log\log k}}$ against the oblivious adversary [3]. The off-line server cost in the above transformation is a lower bound on the shortest path for the robot problem as long as the off-line cost is less than W. If the off-line server cost reaches W, we then just double the window width and restart the server algorithm, each point now corresponding to a larger range of y values. Note that the off-line server cost could be a bit lower than the length of the shortest path since we do not make requests to points corresponding to y-value ranges only partially blocked by obstacles.

As mentioned previously, the on-line cost for the robot is a most the on-line cost for the server problem, plus $W/n + 1$ for each unit advance in the x direction. So, the total distance traveled by the robot is at most $(W_f + n) + W_f 2^{c\sqrt{\log n \log\log n}} \leq 2d(\mathcal{S})2^{c\sqrt{\log n \log\log n}}$. □

6. Non-convex Obstacles

When the obstacles are non-convex, the scene can be a maze. In this case it is easy to see that $\rho(\mathcal{S}, n)$ cannot be bounded by any function of n (the Euclidean distance between s and t). Instead, we prove a ratio between $R(\mathcal{S})$ and $d(\mathcal{S})$ as a function of the total number of vertices in all the obstacles, $|V|$.

Theorem 10: *No randomized algorithm achieves a ratio better than* $|V|/8$.

Proof Outline: Consider the maze in Fig. 3. It has $|V|/8$ passages that could lead from s to t. An algorithm attempts various passages in turn, until it finds the sole passage open to t. For any randomized algorithm, there is one passage whose expected "time to attempt" is $\geq (|V|/8)/2$; this passage is left open to t. The robot walks $2d(\mathcal{S})$ on every failure before that attempt. □

The bound applies *a fortiori* to deterministic algorithms. Rao *et al.* [19] give a deterministic algorithm that explores a maze by building a map of the scene, proceeding at each step to that unexplored vertex of the maze that is nearest to those vertices that have already been visited. It is easy to show that this algorithm achieves a ratio of at most $2|V|$, matching the above lower bound to within a constant. This algorithm is memory intensive, and this may be a handicap when space is limited or the scene changes quickly enough that a map is not worth building. In the full paper we offer a simple, memoryless randomized alternative.

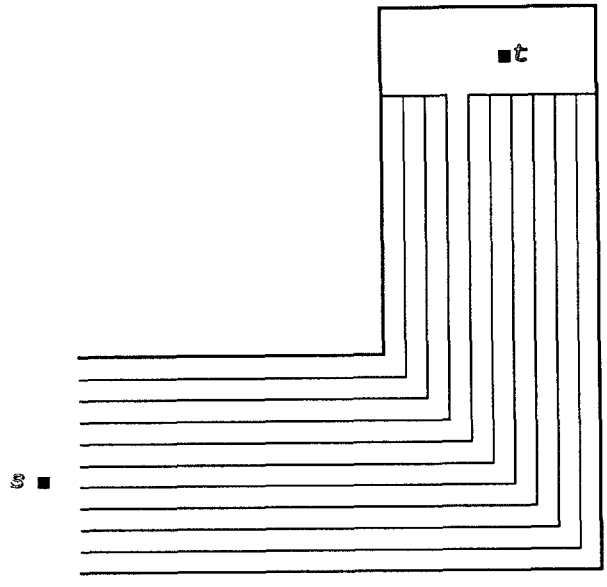

Figure 3: A maze achieving the lower bound

7. Open Problems

- What is the tight bound for the room problem with oriented rectangular obstacles?
- We use the "minimum thickness" requirement — that each obstacle be large enough that a unit circle can be inscribed in it — in bounding the cost of Step 1 of our algorithm for the room problem. Can we dispense with this requirement for the case of oriented rectangular obstacles? (By Theorem 3, we know that this cannot be done for non-oriented rectangular obstacles.) This may be possible if we can give an algorithm achieving a ratio that is a constant, independent of n.
- Can a randomized algorithm for the room problem beat deterministic algorithms?
- Extend the sweep algorithm for the wall problem to handle arbitrary polygonal obstacles,

and hence obtain an algorithm for point-to-point navigation with these obstacles.

- Extend all of the above to three dimensions.
- Give an algorithm that achieves a provably good ratio for three-dimensional scenes with non-convex obstacles (three-dimensional mazes).

Acknowledgements

We thank Alok Aggarwal, Allan Borodin, Don Coppersmith, Leo Guibas, Sandy Irani, Ming Kao, Howard Karloff, Samir Khuller and Yishay Mansour for comments and suggestions.

References

[1] R.A. Baeza-Yates, J.C. Culberson, and G.J.E. Rawlins. Searching in the plane. To appear in *Information and Computation*, 1990.

[2] S. Ben-David, A. Borodin, R.M. Karp, G. Tárdos, and A. Wigderson. On the power of randomization in on-line algorithms. In *Proc. 22nd Annual ACM Symposium on Theory of Computing*, pages 379–388, 1990.

[3] A. Blum, A. Borodin, D. Foster, H.J. Karloff, Y. Mansour, P. Raghavan, M. Saks, and B. Schieber. Randomized on-line algorithms for graph closures. Personal communication, 1990.

[4] M. Blum and D. Kozen. On the power of the compass (or, why mazes are easier to search than graphs). In *19th Annual Symposium on Foundations of Computer Science*, pages 132–142, October 1978.

[5] A. Borodin, N. Linial, and M. Saks. An optimal online algorithm for metrical task systems. In *Proc. 19th Annual ACM Symposium on Theory of Computing*, pages 373–382, 1987.

[6] L. Cheng and J.D. McKendrick. Autonomous knowledge based navigation in an unknown two dimensional environment with convex polygonal obstacles. In *Proc. of the Int. Soc. Opt. Eng.*, volume 1095, pages 752–759, 1989.

[7] E.G. Coffman and E.N. Gilbert. Paths through a maze of rectangles. In preparation, AT&T Bell Labs, 1991.

[8] M. Daily, J. Harris, D. Keirsey, D. Olin, D. Payton, K. Reiser, J. Rosenblatt, D. Tseng, and V. Wong. Autonomous cross-country navigation with the ALV. In *Proc. of the IEEE International Conference on Robotics and Automation*, volume 2, pages 718–726, 1988.

[9] J. Hallam, P. Forster, and J. Howe. Map free localisation in a partially moving 3-D world: the Edinburgh feature based navigator. In *Proc. Intl. Conf. Intelligent Autonomous Systems*, volume 2, pages 726–736, 1989.

[10] B. Kalyanasundaram and K. Pruhs. Visual searching and mapping. Technical Report 90-15, Dept. of Computer Science, Univ. of Pittsburgh.

[11] H.J. Karloff, Y. Rabani, and Y. Ravid. Lower bounds for randomized server algorithms. In *Proc. 23rd Annual ACM Symposium on Theory of Computing*, 1991.

[12] V. Lumelsky. Algorithmic issues of sensor-based robot motion planning. In *26th IEEE Conference on Decision and Control*, pages 1796–1801, 1987.

[13] V.J. Lumelsky and A.A. Stepanov. Dynamic path planning for a mobile automaton with limited information on the environment. *IEEE Transactions on Automatic Control*, AC-31:1058–1063, 1986.

[14] M.S. Manasse, L.A. McGeoch, and D.D. Sleator. Competitive algorithms for on-line problems. *Journal of Algorithms*, 11:208–230, 1990.

[15] H. Moravec. The Stanford cart and the CMU rover. *Proceedings of the IEEE*, 71:872–874, 1983.

[16] B.J. Oomen, S.S. Iyengar, N.S.V., Rao, and R.L. Kashyap. Robot navigation in unknown terrains using learned visibility graphs. Part I: The disjoint convex obstacle case. *IEEE Journal of Robotics and Automation*, 3:672–681, 1987.

[17] C.H. Papadimitriou and M. Yanakakis. Shortest paths without a map. *Proc. 16th ICALP*, pages 610–620, July 1989.

[18] N.S.V. Rao. Algorithmic framework for learned robot navigation in unknown terrains. *IEEE COMPUTER*, 22:37–43, 1989.

[19] N.S.V. Rao, S.S. Iyengar, and G. deSaussure. The visit problem: visibility graph based solution. In *IEEE International Conference on Robotics and Automation*, pages 1650–1655, 1988.

[20] N.S.V. Rao, S.S. Iyengar, B.J. Oomen, and R.L. Kashyap. On terrain model acquisition by a point robot amid polyhedral obstacles. *IEEE Journal of Robotics and Automation*, 4:450–455, 1988.

[21] C.N. Shen and G. Nagy. Autonomous navigation to provide long distance surface traverses for Mars rover sample return mission. In *IEEE International Symposium on Intelligent Control*, pages 362–367, 1989.

[22] D.D. Sleator and R.E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28:202–208, February 1985.

[23] C-K. Yap. Algorithmic motion planning. In J.T. Schwartz and C.K. Yap, editors, *Advances in Robotics*, pages 95–144. Lawrence Erlbaum Associated, Hillsdale, NJ, 1987.