SCHOOL OF OPERATIONS RESEARCH
AND INDUSTRIAL ENGINEERING
COLLEGE OF ENGINEERING
CORNELL UNIVERSITY
ITHACA, NEW YORK 14853

TECHNICAL REPORT NO. 937

October 1990

ALGORITHMIC APPROACHES TO OPTIMAL
ROUTE PLANNING

By

Joseph S. B. Mitchell

To appear: *Proc. SPIE Conference on Mobile Robots*, November 4–9, 1990.

Algorithmic Approaches to Optimal Route Planning

Joseph S. B. Mitchell*

School of Operations Research and Industrial Engineering

Cornell University

Ithaca, NY 14853

Email: jsbm@gvax.cs.cornell.edu

October 5, 1990

Abstract

The field of computational geometry has addressed many problems that are of direct relevance in route planning problems for mobile robots. Examples include the computation of visibility graphs, shortest paths among obstacles, minimal time paths through varied terrain, and various other mission planning problems. In this paper, we survey some recent results and show some applications to computing optimal paths. Our goal is to provide precise geometric models and to analyze algorithms according to their worst-case asymptotic complexities. Our hope is that many of the techniques can be applied as “black boxes” within the framework of a more complex, heuristic-based system to solve the real-world problem.

1 Introduction

The formal study of algorithms to solve geometric problems is very young. The field we call “computational geometry” has its origins in the mid 1970’s, with the pioneering work of M. Shamos [76]. In the last 15 years, a tremendous explosion of results has occurred in this new field, prompted largely by the wealth of applications of geometric algorithms to practical problems in robotics, vision, graphics, and many other engineering disciplines.

The goal of this paper is to survey some of the methods of computational geometry that are relevant to route planning problems for mobile robots. Along the way, we propose several important problem areas for further study. We conclude by suggesting a method

*Partially supported by a grant from Hughes Research Laboratories, Malibu, CA, and by NSF Grant ECSE-8857642.

of applying some of the algorithms discussed to the problem of navigating a mobile robot in an uncertain (or unknown) terrain.

For background material in computational geometry, we refer the reader to the reference texts: Mehlhorn [45], Preparata and Shamos [72], Edelsbrunner [21], O’Rourke [67]. For a more extensive bibliography on shortest path problems, see Mitchell [52].

2 Shortest Obstacle-Avoiding Paths

The most basic kind of geometric shortest path problem is that of finding a shortest path for a point robot in a plane that is cluttered with a set of (known) polygonal obstacles $\mathcal{O}$. We assume that there are h polygonal obstacles, with a total of n vertices describing them. Usually, we also assume that the obstacles are pairwise-disjoint. The restrictive assumption we are making is that the obstacles are both *known* and known *exactly*.

Remarks. (1). The assumption that the obstacles be given as polygons can be lifted — we can instead work with obstacles that have curved arc boundaries (e.g., “splinegons” [79, 46]), provided that certain primitive operations be computable. (2). An alternative means of modeling the problem would be to model free space by giving a regular tessellation of it (e.g., as a set of pixels in a grid, or a quadtree). See Mitchell [50] for a discussion of polygonal versus grid maps in path planning.

If we are interested only in finding a *feasible* obstacle-avoiding path for a disk moving from s to t , then such a path can be found in time $O(n \log n)$ by the “retraction” method of motion planning [84]: compute the Voronoi diagram of the free space in time $O(n \log n)$ (e.g., using Yap’s algorithm [85]), and then search for a feasible path by a linear-time depth-first search on the graph whose edges are the Voronoi edges labeled with the minimal clearances to obstacles. While the path produced in this way is of maximal clearance (since it follows the Voronoi edges), it will not be a shortest path in general.

Once we impose some measure of “goodness” to a path, then we may be interested in the harder problem of computing an *optimal* path. Various measures of optimality are possible, but the most common single measure is the Euclidean length of the path.

If we are searching for a shortest path according to the Euclidean metric, then it is well-known [39,42,78] that a shortest path must be a “taut-string” path that lies on the “visibility graph” (described below) of the obstacle space. Searching the visibility graph for a shortest path is relatively straightforward by using Dijkstra’s algorithm (worst-case time $O(E_{VG} + n \log n)$ [27], for a graph with E_{VG} edges and n nodes) or the A^* algorithm [61] (which has the same worst-case time complexity, but should yield better average-case behavior). The bottleneck in most algorithms for computing shortest paths is in the construction of the visibility graph. Fortunately, some recent progress in computational geometry has yielded some very nice results in the construction of visibility graphs. Below (section 2.1) we give a brief survey. A more detailed discussion is available in the survey article of Alt and Welzl [1].

Rather than computing the VG and searching it, there are other methods too for computing shortest paths. These are discussed in section 2.2 below.

2.1 Visibility Graphs

Given a set of pairwise-disjoint simple polygonal obstacles in the plane, the *visibility graph* (VG) is a graph whose nodes are the set of obstacle vertices and whose edges join pairs of vertices that “see” each other (in that the line segment joining the vertices does not intersect the interior of any obstacle). A VG will have n nodes (one per vertex) and E_{VG} edges, where $E_{VG} \leq \binom{n}{2}$.

The most straightforward algorithm for computing the VG is to check each pair of vertices (u, v) for visibility by testing for intersection between $\overline{uv}$ and each of the segments bounding an obstacle. This method runs in time $O(n^3)$.

A more sophisticated algorithm can be obtained by noting that the *visibility profile* (or *visibility polygon*) about any one vertex v can be found in time $O(n \log n)$ by simply sweeping a ray $r(\theta)$ angularly about v , keeping track of the list $S(\theta)$ of obstacle segments intersecting $r(\theta)$. Each time the ray $r(\theta)$ encounters a new vertex (the vertices have been sorted angularly about v), we insert and/or delete a segment from the list $S(\theta)$, and we make any necessary updates to the visibility profile. The cost per update is $O(\log n)$. By computing the visibility profile about each of the n vertices, we find the VG in time $O(n^2 \log n)$, using only $O(n)$ working space.

The state of the art in VG construction remained at the $O(n^2 \log n)$ level until 1985, when Welzl [81] (and, independently, Asano et al [4]) obtained algorithms whose worst-case running times were $O(n^2)$. These new algorithms rely on the trick of mapping the vertices to their “dual” lines, building the *arrangement* of these lines (in optimal time $O(n^2)$ [24]), and then using the information present in the arrangement to read off the sorted order of vertices about each vertex v in total time $O(n^2)$. Thus, the $O(n)$ angular sorts are not independent of each other, as they can be done collectively in total time $O(n^2)$. Once the angular order is known for vertices about every other vertex, a further trick is necessary to produce the VG without the logarithmic overhead per pair — for example, Welzl [81] uses a topological sort (available from the arrangement) to guide the construction of the visibility profiles about every vertex.

With the notion of topological sweep [22], one was able to find the VG in time $O(n^2)$ using only $O(n)$ working storage.

Output-Sensitive Methods. In the worst case, we know that it takes quadratic time ($O(n^2)$) to compute a visibility graph, since visibility graphs exist with this size. In some cases, however, the visibility graph is very sparse (linear in size). Thus, ideally, we would like an algorithm whose running time is *output-sensitive*, taking time proportional to the size (E_{VG}) of the output.

In the recent work of Ghosh and Mount [29], such an algorithm has been developed with running time $O(n \log n + E_{VG})$ and space requirement $O(E_{VG})$. Independently, Kapoor and Maheshwari [35] obtained a similar bound, but they also were able to reduce the complexity somewhat. Their algorithm requires time $O(n \log n + E_{SP})$ and space $O(E_{SP})$, where E_{SP} is the size of the subgraph of the VG that is relevant for shortest path planning. (In other

words, only those edges of the VG that appear along some nontrivial shortest path are actually discovered and output.) Another related result of Overmars and Welzl [68] shows that the VG can be constructed in time $O(E_{VG} \log n)$ with only $O(n)$ working storage. It remains open whether or not an algorithm exists that requires time $O(n \log n + E_{VG})$ and space $O(n)$. It also remains open whether or not an algorithm exists that matches the lower bound of $\Omega(h \log h + E_{VG})$.

Dynamic Methods. If a new obstacle is added to a scene, one would like to avoid recomputing the VG from scratch. This issue is particularly important when navigating through uncertain terrain, since many or most of the obstacles encountered will not have been known in advance.

A recent technique that is very efficient at adding the effect of a new obstacle to an existing VG has been developed [58]. Consider what happens when a new line segment obstacle $\overline{uv}$ is added to an existing scene. We assume that $\overline{uv}$ is disjoint from the existing obstacles. We first compute the new VG edges that must be added incident to u and v — this can be done in time $O(n \log n)$ by two radial sweeps to compute the visibility profiles about u and v , or can be done in time $O(n)$ using more sophisticated techniques [4]. Next, we need to identify all existing VG edges that cross $\overline{uv}$, since these must be “destroyed”. This is done by a very simple depth-first search in a graph $\mathcal{G}$ whose nodes correspond to edges of VG. Each node of $\mathcal{G}$ is connected to only a small (constant) number of neighbors, and we show that the nodes of $\mathcal{G}$ corresponding to red VG edges form a connected subgraph of $\mathcal{G}$. This allows us to find the red nodes in time proportional to their number. In other words, a new obstacle segment can be added to an existing VG in time $O(n + K)$, where K is the size of the change in the VG. See the full paper for details [58].

An open question we are attempting to address now is the question of efficiently updating a VG when an existing obstacle is *deleted*.

Overlapping Obstacles. All of the methods described so far have assumed that the polygonal obstacles are known to be pairwise-disjoint. If this is not the case, then the only means of applying the existing methods is first to compute the connected components of the obstacle space (e.g., by the line segment intersection search of Chazelle and Edelsbrunner [10]), and then to proceed as before. Unfortunately, the union of all obstacles could have quadratic (in n) complexity, which causes this method to be quartic in the worst case. Alternatively, one can compute the visibility about any single vertex in time $O(n \log n)$ (e.g., using the recent algorithm of Hershberger [31]), yielding an overall time bound of $O(n^2 \log n)$ for constructing the VG and solving the shortest path problem. An even better method can be devised using the recent results of Edelsbrunner et al [23] for finding a single “face” (connected component of free space) in an arrangement of obstacle segments (or polygons). The result is an algorithm that takes time $O(E + n\alpha(n) \log^2 n)$ to find the relevant portion of the VG (of size E) that lies in the connected component of free space that contains the starting point. See Mitchell [53] for details.

2.2 Shortest Path Maps

If we construct the VG and then search it using Dijkstra's algorithm or A^* , we end up with a *shortest path tree* rooted at point s . The tree allows us to trace a shortest path from s to every other vertex. In effect, the tree provides us with a means of answering “vertex queries”, where a query is specified by a point q that must be a vertex, and the answer to the query is the length of a shortest path from s to q (which is answered in constant time) or an explicit listing of a shortest path from s to q (which is answered in time $O(k)$ if the path output has k bends).

A more general data structure is the *shortest path map* (SPM) [40,51], which allows arbitrary queries: given a query point q that lies anywhere within the plane, we can locate q in the SPM in time $O(\log n)$, after which we can return the length of a shortest path from s to q in constant time or an explicit listing of a shortest path from s to q in time $O(k)$ (where k is the number of bends in the path).

We need a few definitions: We say that a vertex r is a *root* of point p if, for some shortest path $\pi(s, p)$, r is the last vertex along $\pi(s, p) \setminus \{p\}$ at which $\pi(s, p)$ bends. If the line segment $\overline{sp}$ is a shortest path, then s is a root of p . The set of all roots of p is denoted by $\mathcal{R}(p)$. (Under a standard “general position assumption”, each vertex p will have a unique root, and every other point p will have at most 3 roots [51].)

The *shortest path map*, $\text{SPM}(s, \mathcal{O})$, with respect to point s and obstacle space $\mathcal{O}$ is a partition of free space $\mathcal{F} = \mathbb{R}^2 \setminus \mathcal{O}$ into maximal regions (called *cells*) that correspond to sets of points with the same root or set of roots with respect to s . More formally, $\text{SPM}(s, \mathcal{O})$ is the partitioning of $\mathcal{F}$ into cells $C(\mathcal{R}) = \{x \in \mathcal{F} \mid \mathcal{R} = \mathcal{R}(x)\}$ corresponding to subsets $\mathcal{R}$ of the set of obstacle vertices. If $\mathcal{R} = \{v\}$ is a singleton vertex, we write $C(v)$ to denote the cell of the SPM *rooted at* vertex v . Each cell $C(v)$ is bounded by obstacle edges, straight segments that are extensions of VG edges, and curved arcs that are subsets of hyperbolas that form the “bisector” curves between pairs of weighted vertices. See Mitchell [51] for a more detailed description of an SPM.

The SPM can be built from a shortest path tree in time $O(n \log n)$. Thus, one method of obtaining the SPM is to build the VG, run Dijkstra's algorithm, and then use the resulting tree to construct the SPM. An alternative approach is to build the SPM directly, and avoid the construction of the (quadratic size) VG. This approach leads Reif and Storer [74] to an $O(hn + n \log n)$ time algorithm, and allows Mitchell [51] to obtain an $O(kn \log^2 n)$ time algorithm, where k is a quantity called the “illumination depth” (and is bounded above by the number of obstacles touched by a shortest path). Both of these algorithms use only linear ($O(n)$) space. The obvious open problem in this area is whether or not an optimal time ($O(n \log n)$) algorithm exists for constructing the SPM.

In mobile robot path planning, one may want to construct $\text{SPM}(t, \mathcal{O})$ corresponding to the *destination* point t . Then, should the robot wander off course at any time, a single logarithmic-time query into $\text{SPM}(t, \mathcal{O})$ will determine the direction to head to go towards the destination along a shortest path. Also, as new obstacles are discovered or old obstacles deleted, one can reasonably efficiently update the SPM accordingly [51].

2.3 Other Metrics

L_1 , L_∞ metrics. If we measure path length by the L_1 or L_∞ metric instead of the Euclidean (L_2) metric, then the “continuous Dijkstra” paradigm of searching for shortest paths while constructing an SPM yields an optimal algorithm requiring $O(n \log n)$ time and $O(n)$ space [49].

Clarkson et al [16] have obtained a nearly optimal algorithm using methods somewhat related to VG methods for Euclidean shortest paths. They show that a sparse grid graph (with $O(n \log n)$ nodes and edges) suffices for searching for shortest paths. This allows them to obtain an $O(n \log^2 n)$ algorithm for the construction and search of this graph.

Fixed Orientations / Approximations. The continuous Dijkstra method of finding L_1 shortest paths generalizes immediately to the case of *fixed orientation metrics* in which distances are measured in terms of the length of the shortest polygonal path whose links are restricted to a set A of fixed orientations. (The L_1 and L_∞ metrics are special cases in which there are 4 fixed orientations, equally spaced by 90 degrees.) The result is an algorithm for finding shortest obstacle-avoiding paths in time $O(kn \log n)$, where $k = |A|$.

We can apply the above result to get a very good approximation algorithm for Euclidean shortest paths by noting that the Euclidean metric is approximated to within accuracy $O(1/k^2)$ by the fixed orientation metric with k equally spaced orientations. The result is an algorithm that runs in time $O((n/\sqrt{\epsilon}) \log n)$ to produce a path guaranteed to have length within factor $(1 + \epsilon)$ of the Euclidean shortest path length [49]. Clarkson [15] also gives an approximation algorithm, using a different method, that produces a path in time $O(n/\epsilon + n \log n)$, after spending $O((n/\epsilon) \log n)$ time to build a data structure of size $O(n/\epsilon)$.

2.4 Other Objectives

Instead of measuring the length of a path as its Euclidean length, several other objective functions are possible, as we describe below.

Minimum Link Paths. Another natural measure of path length is the number of (straight) “links” in the path. The link distance between two points in the plane is the minimum number of links in any obstacle-free polygonal path that joins the two points. A path of minimum link distance may be desirable in applications in which turns are of primary importance.

Suri [80] solved the link distance problem for paths that lie within a single simple polygon; his algorithm requires only linear time in a triangulated simple polygon (and the recent result of Chazelle [9] shows that polygons can be triangulated in linear time). There has also been much related work on link distance problems in simple polygons [36].

Only very recently, though, has the general problem of finding minimum-link paths among obstacles been solved. Mitchell et al [57] give an algorithm whose running time is $O(E_V G \alpha(n) \log^2 n)$, where $\alpha(n)$ is the extremely slowly growing inverse Ackermann function

(which is essentially constant). This result shows that the minimum-link path problem has worst-case time complexity comparable to that of the Euclidean shortest path problem.

A minimum-link path may be far from optimal with respect to Euclidean length; similarly, a shortest Euclidean length path may have thousands of links, while there exists a path joining start and goal that has only 2 links. Thus, a worthy objective in many situations may be to find a path that is both short (in the Euclidean sense) and has few links. No exact method is known for finding the shortest path constrained to have at most k links. This problem and other variations of the “bicriteria” path problem are discussed in the forthcoming paper of Arkin and Mitchell [3].

Minimum Time Paths. Any real mobile robot has a bounded acceleration vector and a maximum speed. If we include these constraints in our model for path planning, then an appropriate objective is to minimize the *time* necessary for a (point) robot to travel from one point of free space to another, with the velocity vector known at the start and possibly constrained at the destination. In general, this kinodynamic planning problem is a very difficult optimal control problem. We are no longer in the nice situation of having optimal paths that are “taut string” paths, lying on a visibility graph. Instead, the paths will be complicated curves in free space, and the complexity of finding such optimal paths remains open.

In a first step towards understanding the algorithmic complexity of computing time-optimal trajectories under dynamic constraints, Canny et al [6] have produced a polynomial-time procedure for finding a *provably good* approximating time-optimal trajectory that is within a factor of $(1 + \epsilon)$ of being a minimum-time trajectory. Their method is fairly straightforward — they discretize the four-dimensional phase space that represents position and velocity. Special care is needed, however, to ensure that the size of the grid is bounded by a polynomial in $1/\epsilon$ and n and the analysis to prove the effectiveness of the resulting paths is quite tedious.

Very recently, Canny et al [7] have provided the first *exact* algorithm for computing an optimal path when there is an upper bound on the L_∞ norm of the velocity and acceleration vectors. Their algorithm is based on characterizing a set of “canonical solutions” (related to “bang-bang” controls in one dimension) that are guaranteed to include an optimal solution path. Then, by writing an appropriate expression in the first-order theory of the reals (see Renegar [75]), they obtain an exponential time, but polynomial space, algorithm. It remains an open question whether or not a polynomial-time algorithm exists.

Bounded Turning Radius. Related to the general problem of handling dynamic constraints, is the important problem of finding shortest paths subject to a bound on their *curvature*. Placing a lower bound on the curvature can be thought of as a means of handling an upper bound on the acceleration vector of a point robot whose speed is constant, or can be thought of as the realistic constraint imposed by the fact that many mobile robots have a bounded steering angle.

Fortune and Wilfong [26] gave an exponential-time decision procedure to determine

whether or not it was possible for a robot to move from a start to a goal among a set of given obstacles, while obeying a lower bound on the curvature of its path (and not allowing reversals). If the point following the path is allowed to reverse direction, then Laumond [37] has shown that it is always possible to obtain a bounded curvature path if a feasible path exists.

Since the general problem seems to be extremely difficult, a restricted version has been studied: Wilfong [82,83] considers the case in which the robot is to follow a given *network* of lanes, with the robot allowed to turn from one segment to another along a (bounded curvature) circular arc if the two lanes intersect. In Wilfong [82], a polynomial-time algorithm is given for producing *some* feasible path; in Wilfong [83], the problem of finding a shortest feasible path is shown to be NP-complete, while a polynomial-time method is given for deforming a given feasible path into a shortest equivalent feasible path. (The time bound is $O(k^3n^2)$, where n is the number of vertices describing the obstacles, and k is the number of turns in the path.)

2.5 Motion of Non-Point Robots

Most of our discussion is focused on the case of point robots. When the robot is not a point, the problem usually becomes much harder. An exception is the case of a circular robot (which is often a very good assumption anyhow) or a non-rotating convex robot. In the case of a circular robot, the problem of finding a shortest path among obstacles is solved almost as in the point robot case — we simply must “grow” the obstacles by the radius of the robot and “shrink” the robot to a point. This is the standard “configuration space” approach in motion planning, and leads to shortest path algorithms with time bounds comparable to the point robot case [5,12,32,48].

Optimal motion of rotating non-circular robots is a very hard problem. Consider the simplest case of moving a line segment (“ladder”) in the plane. The motion planning problem, which ignores any measure of “cost” of motion, is solvable in time $O(n^2 \log n)$ [84]. A natural definition of cost of motion for a ladder is to consider the *work* necessary to move the ladder from one place to another, assuming that there is a uniform coefficient of kinetic friction. Optimal motion of a ladder is an open problem at this point: Papadimitriou and Silverberg [70] and O’Rourke [66] give solutions for restricted cases of moving a ladder among obstacles, and Icking et al. [33] have characterized the solution for the general case *without* obstacles.

2.6 Higher Dimensions

While the shortest path problem in the plane is solvable efficiently, it has been shown that the problem of finding shortest obstacle-avoiding paths according to any L_p ($1 \leq p \leq \infty$) metric in three dimensions is NP-hard [8], even when all of the obstacles are known to be convex polytopes.

The special case in which the path must stay on the surface of a (possibly nonconvex)

polyhedron has been solved in time $O(n^2 \log n)$ [54], and very recently in time $O(n^2)$ [11]. If the obstacles are convex and their number is few, then efficient algorithms are also known [77].

In general, however, the best we know for shortest paths in three dimensions is a fully polynomial approximation scheme [15,69] that produces a path guaranteed to be no longer than $(1 + \epsilon)$ times the length of a shortest path. The time complexity is written in terms of n (the complexity of the obstacles) and ϵ (the user-specified tolerance).

3 Optimal Paths Through Varied Terrain

A natural generalization of the standard shortest obstacle-avoiding path problem is to consider varied terrain in which each region of the plane is assigned a weight that represents the cost per unit distance of traveling in that region. Clearly, the standard problem fits within this framework if we let obstacles have weight ∞ while free space has weight 1.

More formally, we consider the problem in which a planar polygonal subdivision $\mathcal{S}$ is given, with a weight $\alpha \in \{0.1, \dots, W, +\infty\}$ assigned to each face of the subdivision. We let n denote the total number of vertices describing the subdivision. Our objective is to find a path π from s to t that has minimum weighted length over all paths from s to t . (The weighted length of a path is given by the path integral of the weight function — it equals the weighted sum of its Euclidean lengths within each region.) This problem of finding an optimal path within a varied terrain is called the *Weighted Region Problem* (WRP), and was introduced by Mitchell [48] and by Mitchell and Papadimitriou [55].

Applications of the model. There are many applications of the WRP. We originally formulated the WRP to solve the minimum-time path problem for a point robot (without dynamic constraints) moving in a terrain of varied types: grassland, brushland, blacktop, marshland, bodies of water (obstacles to overland travel), and other types of terrain can each be assigned a weight according to the maximum speed at which a mobile robot can traverse the region. In this sense, the weights α denote a “traversability index”, or the reciprocal of maximum speed. Minimizing the weighted length of a path is interpreted as minimizing the time it takes to get from start to goal.

Another application of the WRP that motivated its solution is that of minimizing exposure to “threats” in such a way as to find a *least-risk* path from s to t . In this application, we have knowledge of one or more threats that exist on our terrain (possibly mountainous), and our goal is to minimize some function of the exposure of the path to threats. A simple means of modeling this cost is to assign the weight of a region a value i if there are i threats visible to us while we are in that region. (Also, obstacles get a weight of $+\infty$.) Weights of regions may further be adjusted to reflect the difficulty of traversing the region (as in the minimum-time path formulation).

Finally, an important application of weighted region problems is to the modeling of *uncertainty*. In realistic applications, one does not have an accurate map of exactly where obstacles lie. Even with sensory input data, we cannot always know exactly what lies

ahead of our robot. In such situations, we frequently can only formulate a judgement as far as the *expected* degree of traversability of a certain region. With this interpretation, we can model the situation in which we have incomplete and inaccurate map information that we accumulate while our robot roams through the plane. The weight of a region is the expected cost per unit distance of traveling in that region, based on our a priori assessment of the probability of there being obstacles in the region. Our objective is then to minimize the *expected* length of a path.

The Algorithmic Results. In our paper [55], we present a polynomial-time solution to the WRP that finds a path guaranteed to be within a factor of $(1 + \epsilon)$ of the optimal weighted length, where $\epsilon > 0$ is any user-specified degree of precision. In fact, the algorithm gives a shortest path map that allows efficient queries for ϵ -shortest paths from the source s to *all* other points of the map. The time complexity of the algorithm is $O(E \cdot S)$, where E is the number of “events” in our algorithm and S is the complexity of performing a numerical search of the following form: Given a sequence of k edges through which a path from s to t must go, find a $(1 + \epsilon)$ -shortest path from s to t that goes through this edge sequence. We show that E is bounded above by $O(n^4)$, and there are cases in which E can actually achieve this upper bound (so that no better bound is possible). We also show that the numerical search can be done in such a way that $S = O(k^2 \log(nNW/\epsilon))$, where N is the largest integer coordinate of any vertex of the subdivision $\mathcal{S}$. Since we show that $k = O(n^2)$, this yields an overall time bound of $O(n^8 L)$, where $L = \log(nNW/\epsilon)$ can be thought of as the bit complexity of the problem instance. Although the exponent in our polynomial time bound looks particularly bad, we should emphasize that these are truly *worst-case* bounds (and they are the only polynomial worst-case bounds to our knowledge). In the average case, we expect that E behaves like n or n^2 , and that k is effectively constant.

Special Cases. Several important special cases of the WRP can be solved very quickly by methods other than the general methods of Mitchell and Papadimitriou [55]. In Gewali et al [28], we give quadratic ($O(n^2)$) time algorithms to compute optimal paths when the weights are known to be $\alpha \in \{0, 1, \infty\}$. We obtain similar results even when the weights are allowed to be arbitrary on the edges of the subdivision. For example, if our terrain consists of obstacles ($\alpha = \infty$), relatively cheap regions ($\alpha = 0$), relatively expensive traversible regions ($\alpha = 1$), and an arbitrary network of roads that have various speed limits, then an optimal path can be found in time $O(n^2 \log n)$.

A further extension allows one to compute the *lexicographically* optimal path through weighted regions: minimize the travel in the most expensive (non-obstacle) region; subject to this being minimized, minimize the travel in the second-most-expensive region, etc. A lexicographically optimal path may be a very good starting point in a heuristically-guided search for a shortest path.

4 Other Mission Objectives

We have been discussing route planning problem of the type “get from s to t ”. More complex objectives are frequently encountered. For example, the mission of our robot may be to act as a patrolling “watchman” in a given art gallery. The objective may be to find a shortest cycle (closed route) that allows the robot to “see” all of the gallery. This problem is known as the *watchman route* problem, and has been studied in several papers [13,14,28,62,64]. The general problem is known to be NP-hard, since it is closely related to the *traveling salesman problem* [13,38]. If, however, the art gallery is a simple polygon (without obstacle “holes”), then the problem has a polynomial-time ($O(n^4)$) solution [14]. An interesting open problem in this field is to find fast efficient approximation algorithms to produce provably good routes, such as we know for many versions of the traveling salesman problem [38].

Many other objectives come into play when there is more than one robot. Consider the case of two cooperating robots that must move through a known terrain. In order to stay in communication with each other, there may be a constraint either of the form “stay within distance δ of each other” or “stay visible to each other”. For the second of these constraints, there has been recent progress in solving the problem of finding a *pair* of trajectories (time-parameterized routes) that minimize the sum of the lengths of the two robot paths. We call this problem the “optimal motion of covisible robots” problem, and we have found an $O(E + n \log n)$ time algorithm for its solution [59]. More general versions of problems with many robots are currently being studied. It is also an open problem to plan an optimal pair of trajectories for two robots that must stay within distance δ of each other (without a visibility constraint).

5 Coping With Uncertainty

In any real situation, the terrain map is not known with complete accuracy. There are also inherent inaccuracies in the positional data available for a robot. Thus, we are forced to plan in an uncertain world.

The methodology of *compliant motion planning* has been devised to cope with positional and control uncertainty of the robot [19,25,41]. The classical example of the idea is the “peg-in-hole” scenario in which one wants to exploit compliance in the sliding motion of the peg in contact with the table and the walls of the hole. See Donald [19] for many pointers to the growing literature on compliant motion planning.

We have mentioned already the fact that the weighted region model of terrain can apply to the case of uncertain terrain — the weight on a region is interpreted as the *expected* cost of traversal per unit distance, and the optimal path minimizes the expected length of path. What is missing from this model is the issue of planning for contingencies: for example, once the vehicle reaches a patch of terrain, the cost per unit distance may become known with certainty (e.g., we may discover the presence of an obstacle). This problem needs additional research.

Since map data is inherently inaccurate, it is important to develop representational tools that permit us to store and manipulate maps at various levels of *resolution*, since the resolution of a map can be interpreted as a measure of the certainty with which we know the data. Our goal should be to use a map that represents the data at the precision necessary or reasonable for the application. For example, if we need only a crude path through a maze of complicated obstacles (with, say, a million vertices), then we would like to represent the map with a much sparser map (with, say, a few hundred vertices) that still approximates the original map within a given tolerance ϵ (representing, for instance, the positional uncertainty of our robot). Since the visibility graph is quadratic in the number of vertices, an approximate map can make an enormous savings in computation.

We are thus interested in studying the question of approximating a polygonal subdivision (one type of map representation). A special case of this problem is that of approximating a single polygon: Suppose we are given a simple polygon P with a million vertices. It may be quite complex, with many spirals, and “wiggles”. But if we look at the polygon from a distance, or with eyeglasses that blur our vision, then we will not be able to distinguish the details of the polygon. So, if our problem is such that nothing needs to be any more accurate than a given tolerance ϵ , then we may find that a simple polygon Q of only 100 vertices is more than adequate for representing the original polygon P . We should require Q to stay within “distance” ϵ of P , where the notion of distance needs to be made precise. (One notion of distance between two simple polygons is discussed in Arkin et al [2].) For example, one may consider the case in which the boundary of Q lies within an ϵ -fattening of the boundary of P .

As a first step, we have devised an algorithm that gets close to the optimal answer, giving a simple polygon within an additive error term of the optimal number of sides [30]. We show that the error term is $O(h)$, where h is the number of “holes” in the region defined by the set of all points within distance ϵ of the boundary of the original polygon. We also have addressed the problem of finding approximating planar subdivisions of minimum combinatorial complexity. Again we have an algorithm that is guaranteed to give an answer close to the optimum.

The practical significance of these results is the ability to take a scene, say of many polygons with 10,000 vertices, and simplify it to any desired degree of accuracy, obtaining, say, a new scene consisting of only 100 vertices that is *guaranteed* to be a good approximation to the original scene, within a given ϵ .

The main technical issue that must be addressed is the issue of maintaining a given *topology* of a scene, while minimizing its combinatorial size. For example, one could take an ϵ -fattening of a polygon P and then compute a minimum link-distance (i.e., minimum combinatorial size) cycle that lies within the fattened (gray) region, while surrounding the holes in the gray region. However, this strategy does not preserve the *topology* of the original object (simple polygon P), since the new approximating cycle may not be *simple* – in finding the minimum-link cycle in the gray region, we have ignored the fact that the cycle must be simple. It is not clear how one would go about finding a minimum-link simple cycle.

Instead, our approach is to give up on trying to find an *optimal* approximating polygon, and try to find an approximating polygon that is guaranteed to be simple, while not guaranteed to be optimal (among the set of all simple cycles). Our goal, then, is to show that while giving up on optimality, we have not given up too much. We show that if one takes the minimum-link cycle (ignoring the topological constraint of simplicity), and then “untangles” it in a clever way, it does not cost more than $O(h)$ extra bend points to do the untangling. The result is that we can get a very good approximation to the minimum-size object whose boundary closely follows the original object. The running time of the algorithm is $O(n \log n)$.

6 Finding Routes Without a Map

Many researchers have addressed the problem of planning paths for a mobile robot that does not have a complete map known in advance, but does have on-board sensors of various types [17,20,34,43,44,47,56,60,65,71,73]. A general paradigm for solving path planning problems in the presence of little (or no) map information is as follows:

Dynamic Route Planning Paradigm (DRPP)

- (0) Let $\text{MAP}(0)$ be the map representation at time 0. Let $p(0)$ be the position of the robot at time 0. We plan an optimal path $\pi^* = \text{plan}(p(0), \text{MAP}(0))$, starting at point $p(0)$, using the $\text{MAP}(0)$.
- (1) While there is no other input from sensors, we continue to traverse π^* . Let $p(t)$ be the robot’s position at time t . (Note that $p(t)$ will not in general be known with complete accuracy.)
- (2) If at time t_i we obtain new map information $\text{NEWMAP}(t_i)$, then we let $\text{MAP}(t_i^+) \leftarrow \text{merge}(\text{MAP}(t_i^-), \text{NEWMAP}(t_i))$ be the result of merging the new data in with the old. (The new data $\text{NEWMAP}(t_i)$ presumably comes from some external information source or from our sensor data (e.g., a laser radar, a sonar scanner, or a touch sensor that tells us that we hit an obstacle).) We then set $\pi^* \leftarrow \text{plan}(p(t), \text{MAP}(t_i^+))$, and go to (1).

Clearly, the heart of the problem of planning routes dynamically in unknown or uncertain terrain lies in the description of the *plan* and *merge* procedures of the above outline. Both procedures rely crucially on the choice of MAP representation. If the sensory data is arriving at real-time rates, then it is imperative that the merging be done at similar real-time rates, so we may be forced to examine MAP representations that keep only a small fraction of the geometric data available. For example, each time that we call *merge*, we may need to discard some of the information previously stored in $\text{MAP}(t_i^-)$. This process could be implemented by decreasing over time the resolution with which data (e.g., a planar subdivision) is stored; the resolution with which a portion of the map is stored

may be a function of the *age* of the data (when did our sensors last “see” it?) and of the number of times that we have encountered the data (have we seen this portion of the map over and over again?).

As one specific application of this paradigm using some of the tools we have discussed, let us consider the following problem. Assume that our robot is a point (or circle) moving among a set of unknown obstacles in the plane, and assume that it is equipped with a sensor that allows it to discover obstacles and free space as it roams. Any portion of the plane that is unknown is “gray”; any portion that is known to be an obstacle is “black”; and any portion that is known already to be free space is “white”. At any given instant, our robot sits at a point in white space and has knowledge of a polygonal map with white, black, and gray labeled regions. Each time the robot receives a new piece of data, it modifies the map accordingly and possibly adjusts its current plan.

We can cast this problem into the context of the weighted region problem (WRP) by assigning a weight α to the gray region, where α should indicate our best guess at the traversability of the unknown terrain. (α can be adjusted and estimated based on the relative density of obstacles seen so far.) White regions have weight 1, and black regions have weight infinity. Then, from our given location s , we desire a shortest weighted path to the goal point t . This clearly falls within the domain of the WRP already studied [55]. In this special case, though, it is reasonable to use a particularly fast approximation: Let us make the additional constraint that once we enter the gray region, we must stay in the gray region. In other words, we desire a path of minimum total weight that takes us from our current position s through the known white space, enters the gray region at a point p , and then continues through the gray region to the goal point t (which presumably lies within the gray region as well). With this assumption, we can get an efficient solution as follows: We compute a shortest path map $\text{SPM}(t, G)$ within the gray region G that contains the goal point, using t as a source. This gives us a description of the best path from t to any point on the boundary of G . Similarly, we compute an $\text{SPM}(s, W)$ within the white region W , with the source point being our current vehicle position s . Both of these calculations can be done in time $O(n^2)$ or better (by the output-sensitive methods discussed earlier). We then search for a point p on the common boundary of W and G that minimizes the weighted sum $d(s, p) + \alpha d(p, t)$, where $d(\cdot, \cdot)$ indicates the shortest path distance within the white or the gray region. This search is done in linear time by a simple traversal of the common boundary of G and W . Thus, we have given a fast means of solving the planning problem under this (simplified) model of uncertainty.

7 Conclusion

It is hoped that some of the recent algorithmic tools that we have discussed can be applied to the hard issues in robot route planning, and that other new problems of particular importance in mobile robotics will be brought to the attention of the field of computational geometry.

References

- [1] H. Alt and E. Welzl, "Visibility Graphs and Obstacle-Avoiding Shortest Paths", *Zeitschrift für Operations Research*, Vol. 32, pp. 145-164.
- [2] E.M. Arkin, P. Chew, D.P. Huttenlocher, K. Kedem, J.S.B. Mitchell, "An Efficiently Computable Metric for Comparing Polygonal Shapes", *Proc. First ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, CA, January 22-24, 1990, pp. 129-137. To appear: *IEEE Trans. on Pattern Analysis and Machine Intelligence*.
- [3] E.M. Arkin and J.S.B. Mitchell, "On Geometric Bicriteria Path Problems", Manuscript, 1990.
- [4] T. Asano, T. Asano, L. Guibas, J. Hershberger, and H. Imai, "Visibility of Disjoint Polygons", *Algorithmica*, Vol. 1, (1986), pp. 49-63.
- [5] B. Baker, "Shortest Paths With Unit Clearance Among Polygonal Obstacles", *SIAM Conference on Geometric Modeling and Robotics*, Albany, NY, July 15-19, 1985.
- [6] J. Canny, B. Donald, J. Reif, and P. Xavier, "On the Complexity of Kinodynamic Planning", *Proc. 29th IEEE FOCS*, 1988.
- [7] J. Canny, A. Rege, and J. Reif, "An Exact Algorithm for Kinodynamic Planning in the Plane", *Proc. 6th Ann. ACM Sympos. Comput. Geom.*, 1990, pp. 271-280.
- [8] J. Canny and J. Reif, "New Lower Bound Techniques for Robot Motion Planning Problems", *Proc. 28th FOCS*, pp. 49-60, Oct. 1987.
- [9] B. Chazelle, "Efficient Polygon Triangulation", CS-TR-249-90, Princeton Univ., 1990. To appear: *IEEE FOCS* 1990.
- [10] B. Chazelle and H. Edelsbrunner, "An Optimal Algorithm for Intersecting Line Segments in the Plane", Tech. Report UIUCDCS-R-1419, Department of Computer Science, University of Illinois at Urbana Champaign, March 1988.
- [11] J. Chen and Y. Han, "Shortest Paths on a Polyhedron", *Proc. 6th Annual ACM Symposium on Computational Geometry*, Berkeley, CA, June 6-8, 1990, pp. 360-369.
- [12] L.P. Chew, "Planning the Shortest Path for a Disk in $O(n^2 \log n)$ Time", *Proc. First Annual ACM Symposium on Computational Geometry*, Baltimore, MD, June 1985, pp. 214-220.
- [13] W.P. Chin and S. Ntafos, "Optimum Watchman Routes", *Information Processing Letters* Volume 28, pp. 39-44, May 1988.
- [14] W.P. Chin and S. Ntafos, "Watchman Routes in Simple Polygons", Technical Report, Computer Science, University of Texas at Dallas, 1987. To appear: *Discrete and Computational Geometry*.
- [15] K. Clarkson, "Approximation Algorithms for Shortest Path Motion Planning", *Proc. 19th Annual ACM Symposium on Theory of Computing*, New York City, May 25-27, 1987, pp. 56-65.

- [16] K. Clarkson, S. Kapoor, and P. Vaidya, "Rectilinear Shortest Paths through Polygonal Obstacles in $O(n \log^2 n)$ Time", *Proc. Third Annual ACM Symposium on Computational Geometry*, Waterloo, Ontario, 1987, pp. 251-257.
- [17] A. Datta and K. Krithivasan, "Path Planning with Local Information", *Proc. Foundations of Software Technology and Theoretical Computer Science*, New Delhi, India, December, 1988. ??
- [18] P.J. de Rezende, D.T. Lee, and Y.F. Wu, "Rectilinear Shortest Paths With Rectangular Barriers", *Proc. First Annual ACM Symposium on Computational Geometry*, Baltimore, MD, June 1985, pp. 204-213.
- [19] B.R. Donald, "The Complexity of Planar Compliant Motion Planning under Uncertainty", *Proc. 4th Annual ACM Sympos. Comput. Geom.*, pp. 309-318, 1988.
- [20] P. Eades, X. Lin, and N.C. Wormald, "Performance Guarantees for Motion Planning with Temporal Uncertainty", Manuscript, Dept. of Computer Science, Univ. of Queensland, St. Lucia, Queensland, 1989. Presented at the First Canadian Conference on Computational Geometry, Montreal, Canada, August, 1989.
- [21] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer-Verlag, Heidelberg, Germany, 1987.
- [22] H. Edelsbrunner and L. Guibas, "Topologically Sweeping an Arrangement", *Journal of Computer and System Sciences* (1989), **38**, pp. 165-194.
- [23] H. Edelsbrunner, L.J. Guibas, and Micha Sharir, "The Complexity and Construction of Many Faces in Arrangements of Lines and of Segments", *Discrete and Computational Geometry* **5** (1990), pp. 161-196.
- [24] H. Edelsbrunner, J. O'Rourke, and R. Seidel, "Constructing Arrangements of Lines and Hyperplanes with Applications", *SIAM J. Comput.* (1986), **15**, pp. 341-363.
- [25] M. Erdmann, "Using Backprojections for Fine Motion Planning with Uncertainty", *Int. J. of Robotics Research*, Vol. 5, No. 1 (1986).
- [26] S. Fortune and G. Wilfong, "Planning Constrained Motion", *Proc. 20th ACM STOC* Chicago, May, 1988, pp. 445-459.
- [27] M. Fredman and R. Tarjan, "Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms", *Proc. 25th Annual IEEE Symposium on Foundations of Computer Science*, pp. 338-346, 1984.
- [28] L. Gewali, A. Meng, J.S.B. Mitchell, and S. Ntafos, "Path Planning in $0/1/\infty$ Weighted Regions With Applications", *ORSA Journal on Computing* Vol. 2, No. 3, Summer, 1990, pp. 253-272.
- [29] S.K. Ghosh and D.M. Mount, "An Output Sensitive Algorithm for Computing Visibility Graphs", *Proc. 28th Annual IEEE Symposium on Foundations of Computer Science*, pp. 11-19, 1987. To appear: *Algorithmica*.

- [30] L. Guibas, J. Hershberger, J. Mitchell, and J. Snoeyink, "On Approximating Simple Polygons by Simple Polygons with Fewer Sides". Manuscript in preparation, 1990.
- [31] J. Hershberger, "Finding the Upper Envelope of n Line Segments in $O(n \log n)$ Time", *Information Processing Letters* **33** (1989), pp. 169-174.
- [32] J. Hershberger and L.J. Guibas. "An $O(n^2)$ Shortest Path Algorithm for a Non-Rotating Convex Body", *Journal of Algorithms* **9**, 18-46 (1988).
- [33] C. Icking, G. Rote, E. Welzl, and C. Yap, "Shortest Paths for Line Segments", Technical Report, Fachbereich Mathematik, Freie Universität, Berlin, September, 1989.
- [34] S.S. Iyengar, C.C. Jorgensen, S.V.N. Rao, and Weisbin, "Robot Navigation Algorithms Using Learned Spatial Graphs", *Robotica*, **4**, (1986), pp. 93-100.
- [35] S. Kapoor and S.N. Maheshwari. "Efficient Algorithms for Euclidean Shortest Path and Visibility Problems with Polygonal Obstacles". *Proc. Fourth Annual ACM Symposium on Computational Geometry*, Urbana-Champaign, IL, June 6-8, 1988, pp. 172-182.
- [36] Y. Ke, "An Efficient Algorithm for Link Distance Problems", *Proc. 5th Annual ACM Symposium on Computational Geometry*, 1989, pp. 69-78.
- [37] J. Laumond, "Feasible Trajectories for Mobile Robots with Kinematic and Environment Constraints", In *Intelligent Autonomous Systems*, 1986, pp. 346-354.
- [38] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys (eds.), *The Traveling Salesman Problem*, John Wiley & Sons, New York, 1985.
- [39] D.T. Lee, "Proximity and Reachability in the Plane". Ph.D. Thesis, Technical Report ACT-12, Coordinated Science Laboratory, University of Illinois, Nov. 1978.
- [40] D.T. Lee and F.P. Preparata. "Euclidean Shortest Paths in the Presence of Rectilinear Boundaries", *Networks*, **14** (1984), pp. 393-410.
- [41] T. Lozano-Pérez, M.T. Mason, and R. Taylor, "Automatic Synthesis of Fine-Motion Strategies for Robots", *Int. J. of Robotics Research*, Vol. 3, No. 1 (1984).
- [42] T. Lozano-Pérez and M.A. Wesley, "An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles", *Communications of the ACM*, Vol. 22, No. 10 (1979), pp. 560-570.
- [43] V.J. Lumelsky and A.A. Stepanov. "Path-Planning Strategies for a Point Mobile Automaton Moving Amidst Unknown Obstacles of Arbitrary Shape", *Algorithmica*, **2**, (1987), pp. 403-430.
- [44] V.J. Lumelsky, S. Mukhopadhyay, and K. Sun. "Dynamic Path Planning in Sensor-Based Terrain Acquisition", *IEEE Transactions on Robotics and Automation*, Vol. 6, No. 4, August, 1990, pp. 462-472.
- [45] K. Mehlhorn, *Multi-dimensional Searching and Computational Geometry*, EATCS Monographs on Theoretical Computer Science, Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, 1984.

- [46] E.A. Melissaratos and D.L. Souvaine, "Shortest Paths, Visibility, and Optimization Problems in Planar Curvilinear Objects", Extended Abstract in *Proc. 2nd Canadian Conference on Computational Geometry*, Ottawa, Ontario, August 6-10, 1990, pp. 337-342.
- [47] J.S.B. Mitchell, "An Autonomous Vehicle Navigation Algorithm", *Proc. SPIE Applications of Artificial Intelligence*, Vol. 485, pp. 153-158, 1984.
- [48] J.S.B. Mitchell, "Planning Shortest Paths", PhD Thesis, Department of Operations Research, Stanford University, August, 1986.
- [49] J.S.B. Mitchell, "An Optimal Algorithm for Shortest Rectilinear Paths Among Obstacles", Manuscript, 1988. Presented at the First Canadian Conference on Computational Geometry, Montreal, Canada, August, 1989.
- [50] J.S.B. Mitchell, "An Algorithmic Approach to Some Problems in Terrain Navigation", *Artificial Intelligence*, **37** (1988) pp. 171-201.
- [51] J.S.B. Mitchell, "A New Algorithm for Shortest Paths Among Obstacles in the Plane", Technical Report No. 832, School of Operations Research and Industrial Engineering, Cornell University, October, 1988. To appear: *Annals of Mathematics and Artificial Intelligence*.
- [52] J.S.B. Mitchell, "A Bibliography on (Geometric) Shortest Path Planning", Manuscript, School of Operations Research and Industrial Engineering, Cornell University, July, 1990. (This is an ongoing project and will be updated as new papers appear.)
- [53] J.S.B. Mitchell, "Computing Visibility Graphs of Intersecting Obstacles", Manuscript, 1990.
- [54] J.S.B. Mitchell, D.M. Mount, and C.H. Papadimitriou, "The Discrete Geodesic Problem", *SIAM Journal on Computing*, **16**, No. 4, pp. 647-668, August, 1987.
- [55] J.S.B. Mitchell and C.H. Papadimitriou, "The Weighted Region Problem: Finding Shortest Paths Through a Weighted Planar Subdivision", Technical Report No. 885, School of Operations Research and Industrial Engineering, Cornell University. To appear: *Journal of the ACM*.
- [56] J.S.B. Mitchell, D.W. Payton, and D.M. Keirsey, "Planning and Reasoning for Autonomous Vehicle Control", *International Journal of Intelligent Systems*, **II**, (1987), pp. 129-198.
- [57] J.S.B. Mitchell, G. Rote, and G. Woeginger, "Minimum-Link Paths Among Obstacles in the Plane", *Proc. 6th Annual ACM Symposium on Computational Geometry*, Berkeley, CA, June 6-8, 1990, pp. 63-72.
- [58] J.S.B. Mitchell and E. Welzl, "Dynamically Maintaining a Visibility Graph under Insertions of New Obstacles", Manuscript, Cornell University, 1989.
- [59] J.S.B. Mitchell and E.L. Wynters, "Optimal Motion of Covisible Points Among Obstacles in the Plane", Extended Abstract in *Proc. 2nd Canadian Conference on Computational Geometry*, Ottawa, Ontario, August 6-10, 1990, pp. 116-119. Full paper available as Technical Report 896, School of Operations Research and Industrial Engineering, Cornell University, March, 1990.

- [60] B.G. Mobasseri, "Path Planning Under Uncertainty from a Decision Analytic Perspective", *Proc. IEEE International Symposium on Intelligent Control*, Albany, NY, Sept. 25-26, 1989, pp. 556-560.
- [61] N.J. Nilsson, *Principles of Artificial Intelligence*, Tioga Publishing Co., Palo Alto, CA, 1980.
- [62] B.J. Nilsson and D. Wood, "Optimum Watchmen Routes in Spiral Polygons: Extended Abstract", Manuscript, Univ. of Waterloo, Ontario, Canada, August, 1989.
- [63] S. Ntafos, "The Robber Route Problem", Technical Report, Computer Science, University of Texas at Dallas, 1988.
- [64] S. Ntafos and L. Gewali, "External Watchman Routes", Technical Report, Computer Science, University of Texas at Dallas, 1989.
- [65] B.J. Oommen, S.S. Iyenger, S.V. Rao, and R.L. Kashyap, "Robot Navigation in Unknown Terrains Using Learned Visibility Graphs. Part I: The Disjoint Convex Obstacle Case", Technical Report SCS-TR-86, School of Computer Science, Carleton University, February 1986.
- [66] J. O'Rourke, "Finding a Shortest Ladder Path: A Special Case", Technical Report, Institute for Mathematics and its Applications, Preprint Series No. 353, University of Minnesota, October 1987.
- [67] J. O'Rourke, *Art Gallery Theorems and Algorithms*, Oxford University Press, 1987.
- [68] M.H. Overmars and E. Welzl, "New Methods for Computing Visibility Graphs", *Proc. Fourth Annual ACM Symposium on Computational Geometry*, Urbana-Champaign, IL, June 6-8, 1988, pp. 164-171.
- [69] C.H. Papadimitriou, "An Algorithm for Shortest-Path Motion in Three Dimensions", *Information Processing Letters*, **20** (1985), pp. 259-263.
- [70] C.H. Papadimitriou and E.B. Silverberg, "Optimal Piecewise Linear Motion of an Object Among Obstacles", *Algorithmica*, **2**, 1987, pp. 523-539.
- [71] C.H. Papadimitriou and M. Yannakakis, "Shortest Paths Without a Map (Extended Abstract)", Manuscript, UCSD, November 1988. Appears in *Proc. ICALP 89*.
- [72] F.P. Preparata and M.I. Shamos, *Computational Geometry*, Springer-Verlag, New York, Berlin, Heidelberg, Tokyo, 1985.
- [73] N.S.V. Rao, S.S. Iyengar, C.C. Jorgensen, and C.R. Weisbin, "Robot Navigation in an Unexplored Terrain", *Journal of Robotic Systems*, **3**, No. 4, (1986), pp. 389-407.
- [74] J.H. Reif and J.A. Storer, "Shortest Paths in Euclidean Space with Polyhedral Obstacles", Technical Report CS-85-121, Computer Science Department, Brandeis University, April, 1985.
- [75] J. Renegar, "On the Computational Complexity and Geometry of the First Order Theory of the Reals", Technical Report No. 853, SORIE, Cornell University, 1989.

- [76] M.I. Shamos, "Computational Geometry", Ph.D. Thesis, Dept. of Computer Science, Yale University (1978).
- [77] M. Sharir, "On Shortest Paths Amidst Convex Polyhedra", Technical Report No. 181, Robotics Report No. 53, New York University, Dept. of Computer Science, Courant Institute of Mathematical Sciences, September, 1985.
- [78] M. Sharir and A. Schorr, "On Shortest Paths in Polyhedral Spaces", *SIAM Journal on Computing* Vol. 15, No. 1, pp. 193-215, February 1986.
- [79] D. Souvaine, "Computational Geometry in a Curved World", Ph.D. Thesis, Technical Report CS-TR-094-87, Department of Computer Science, Princeton University, October, 1986.
- [80] S. Suri, "A Linear Time Algorithm for Minimum Link Paths inside a Simple Polygon", *Computer Vision, Graphics, and Image Processing*, **35**, (1986), pp. 99-110.
- [81] E. Welzl, "Constructing the Visibility Graph for n Line Segments in $O(n^2)$ Time", *Information Processing Letters*, Vol. 20 (1985), pp. 167-171.
- [82] G. Wilfong, "Motion Planning for an Autonomous Vehicle", *IEEE Int. Conf. Robotics and Automation*, 1988, pp. 529-533.
- [83] G. Wilfong, "Shortest Paths for Autonomous Vehicles", Manuscript, AT& T Bell Labs, October, 1988.
- [84] C.K. Yap, "Algorithmic Motion Planning", in *Algorithmic and Geometric Aspects of Robotics*, Vol. 1, (J.T. Schwartz and C. Yap, Ed.), Lawrence Erlbaum Associates, Hillsdale, NJ, 1987, pp. 95-144.
- [85] C.K. Yap, "An $O(n \log n)$ Algorithm for the Voronoi Diagram of a Set of Simple Curve Segments", *Discrete and Computational Geometry* **2** (1987), pp. 365-393.