# An Algorithmic Approach to Some Problems in Terrain Navigation

**Joseph S.B. Mitchell**
*School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY 14853, U.S.A.*

ABSTRACT

*Recent advances in the field of computational geometry have provided efficient algorithms for a variety of shortest path problems. Many problems in the field of terrain navigation can be cast as optimal path problems in a precise geometric model. With such a model one can develop and analyze algorithms for the solution of the original problem and can gain insights into how to design more efficient heuristics to deal with more complex problems. We examine the path planning problem in which we are given a "map" of a region of terrain and we are expected to find optimal paths from one point to another. This, for example, is a task which must be done repeatedly for the guidance of an autonomous vehicle. We examine how to formulate some path planning problems precisely, and we report algorithms to solve certain special cases.*

## 1. Introduction

One of the basic tasks required of an autonomous vehicle is to move from one location to another. Usually, the robot has some knowledge (which we refer to as a *map*) of the terrain on which it is navigating, although it may initially have none. It should try to exploit as much of this knowledge as possible in order to make the best decisions about its path. Two types of navigation problems immediately arise: How does the robot *locate* its position in the map, and how does one use the map and the current location of the robot to *plan* a route to the goal? In this paper, we concentrate on the latter type of problem. We use the word "terrain" to refer to both indoor and outdoor environments for a vehicle.

The problem of location involves using sensory data to compute the coordinates of the robot. This can be done only to within some error bounds; however, we will assume that the errors involved are small enough in comparison with the scale of the planning problem that we can assume the exact position of the robot is known at each point in time and that the robot can be

*Artificial Intelligence* **37** (1988) 171–201

controlled exactly to follow a specified trajectory. The issue of location uncertainty is very important, and it is an area of continuing research.

In planning paths through the map, we typically wish to minimize some objective function which specifies the *cost* of motion along the path (the *path length*). Depending on how the terrain is modeled and what the optimization criteria are, this results in various versions of the *shortest path problem*. The shortest path problem for a mobile robot was one of the early problems addressed by researchers in artificial intelligence (see Nilsson [59] and Moravec [58]).

In this paper we discuss some of the issues that go into selecting a representation of a map, and we report some of the recent results from the field of *computational geometry* which are of use in solving terrain navigation problems. Computational geometry and algorithmic motion planning have recently erupted as important areas of research in computer science. We certainly cannot do justice here to the field of motion planning, so we refer the interested reader to the superb new books of Schwartz and Yap [78] and Schwartz et al. [77], which collect together many of the landmark papers of this exciting field. We also refer the reader to the award-winning thesis of Canny [9]. Much of our discussion uses concepts from the field of computational geometry. We refer the interested reader to three recent books on the subject: Edelsbrunner [19], Mehlhorn [44], and Preparata and Shamos [65].

While many of the problems of terrain navigation have been addressed before from a heuristic or approximation point of view, we feel that it is important to formalize some of the models and to discuss the computational complexity of exact algorithms. We have found that an understanding of the underlying geometry can lead to new heuristics and faster algorithms for a variety of problems.

How do we go about representing varied terrain? What representations yield particularly natural or efficient algorithms for path planning? What do shortest paths "look like"; that is, can we understand enough about the local optimality criteria to cut down the search for optimal paths?

We begin addressing some of these issues in the following sections, where we describe the *obstacle avoidance problem*, the *discrete geodesic problem*, the *weighted region problem*, and some special cases of these problems that arise naturally in finding shortest paths through terrain. Throughout our discussions, we concentrate on the fundamental algorithmic issues of computing paths, and we are interested in worst-case asymptotic running times. Where appropriate, we will allude to the practicality of the algorithms and approaches we suggest.

## 2. Obstacle Avoidance in the Plane

The simplest surface to model is a plane, and the simplest nontrivial terrain map on a plane is a binary partitioning of the plane into regions that are

obstacles and regions that are traversable by the vehicle. Usually, the robot can be modeled as a point, which is a reasonable assumption if it is small in comparison with the dimensions of the terrain map. Also, by *configuration space* techniques, many problems of moving a nonpoint robot among obstacles can be reduced to the case of moving a point (possibly in higher dimensions), by "growing" the obstacles in an appropriate way, as in [41]. The problem of planning a shortest Euclidean path from one point to another is then the usual *obstacle avoidance shortest path problem*. This problem has been studied by many researchers over the last twenty years. A survey of many algorithms is given in [49, 55].

Some recent work of Papadimitriou and Silverberg [64] and O'Rourke [61] has started to address the tough problem of *shortest* paths for a nonpoint noncircular body in the plane. While this can be mapped into a shortest path problem in three-dimensional configuration space, calculating shortest paths in three dimensions has been shown to be quite difficult, as we will mention later.

There are two standard approaches to modeling the terrain for the obstacle avoidance problem. The first subdivides the plane into small regular pieces ("pixels") and labels each piece as obstacle or nonobstacle. If all of the pieces are squares of exactly the same size, then we get the standard grid tesselation, and we can represent the map as a binary array. A measure of the size of the problem instance is given by $n$, the total number of pixels. Of course, $n$ depends on the resolution of the grid used to approximate the set of obstacles. Picking a very small pixel (very high resolution) allows a better model of the obstacle space, but it costs us significantly in problem size. If using a regular grid, we may be forced to have pixels smaller than a certain size in order to assure that skinny passage ways between obstacles or tiny little obstacles appear in the map. Other subdivision representations exist which are hierarchical in nature, such as the *quadtree* (see Samet and Webber [76] or Kambhampati and Davis [31]). These schemes have the advantage that they conglomerate groups of contiguous small squares into single larger squares.

Another standard approach to modeling terrain which is obstacle or nonobstacle is to give a description of a polygonal space in terms of its boundary representation. Obstacles are given as a list of $k$ simple polygons, each represented by a doubly linked list of vertices (each vertex just being a pair of coordinates, either integer or real). Usually, the obstacles are assumed to be disjoint. Let $n$ be the total number of vertices of all polygonal obstacles. We refer to $n$ as the "complexity of the scene", as it is proportional to the number of bytes of storage needed to represent the map. To be more precise, though, the complexity of the scene should be the pair $(n, k)$, since some of the complexities of algorithms depend not only on $n$, but also on $k$. (Of course, $k$ is always less than or equal to $n$, but it may be significantly less than $n$, in which case it is important to know precisely how the running times depend on $k$). The objective of the algorithmic study of the shortest path problem is to determine

an algorithm whose worst-case running time is a low-degree polynomial in $n$ and $k$. Most work in computational geometry focuses on *worst-case* running times; however, it is an important issue (not addressed here) to consider the *average-case* running times of shortest path algorithms.

An example of a set of obstacles is given in Fig. 1. In Fig. 1(a), we show the representation of the obstacles as a set of $k = 2$ polygons with $n = 8$ vertices and the shortest path from start to goal. Then, in Fig. 1(b) we show the digitized representation on a grid of $16 \times 16$ pixels ($n = 256$). In Fig. 1(c) we show the same set of obstacles represented as a quadtree, where the free space is also subdivided into a quadtree representation. The total size of the representation in this case is $n = 124$.

There are other methods of modeling the terrain in the obstacle avoidance problem. The obstacles may be given as a list of circles, ellipses, or other nonpolygonal objects. The shortest path problem among circular or elliptical obstacles has been addressed by Baker [5], Chew [11], and Mitchell [49]. Souvaine [81] introduced the notion of *splinegons* as a means of describing simple closed *curved* figures, and she describes many generalizations of standard computational geometry algorithms to the case of splinegons. A different approach to modeling obstacle maps is given by Brooks [7] and Brooks and Lozano-Pérez [8], who give a decomposition of the free space into generalized cylinders. Their (heuristic) method is similar to the retraction method of motion planning (see Schwartz and Yap [78]).

Shortest paths according to the Euclidean metric will be "taut string" paths: imagine threading a string from START to GOAL among the obstacles, then pulling the string until it is taut. The question really comes down to how the string should be threaded. One cannot afford to try all of the exponentially many possible threadings.

The algorithm one uses to find shortest paths among obstacles depends on the representation used in the map. For the case of a binary grid representation, the search for shortest paths is easy and straightforward. Shortest paths are found by restricting the robot's motion to connect centers of adjacent free squares. Then, we can consider the graph (so-called *grid graph*) whose nodes are the free space squares and the edges connect adjacent free squares. The degree of a node is thus at most 8, so the graph has at most $n$ nodes and no more than $8n$ edges. Lengths are assigned to edges in the obvious way: diagonal edges are of length $\sqrt{2}$ times the size of a square, and horizontal and vertical edges are of length equal to the size of a square. We can then search the graph for shortest paths by applying the A* algorithm (Nilsson [59]) or the Dijkstra [18] algorithm (which is "uniformed" A*). The worst-case running time of Dijkstra's algorithm (and hence of A*) is $O(|E| + |V|\log|V|)$ for a graph with $|E|$ edges and $|V|$ nodes (Fredman and Tarjan [21]); thus, the worst-case running time of the shortest path algorithm is $O(n \log n)$ for a grid with $n$ pixels. Note that for A* there is an obvious admissible heuristic

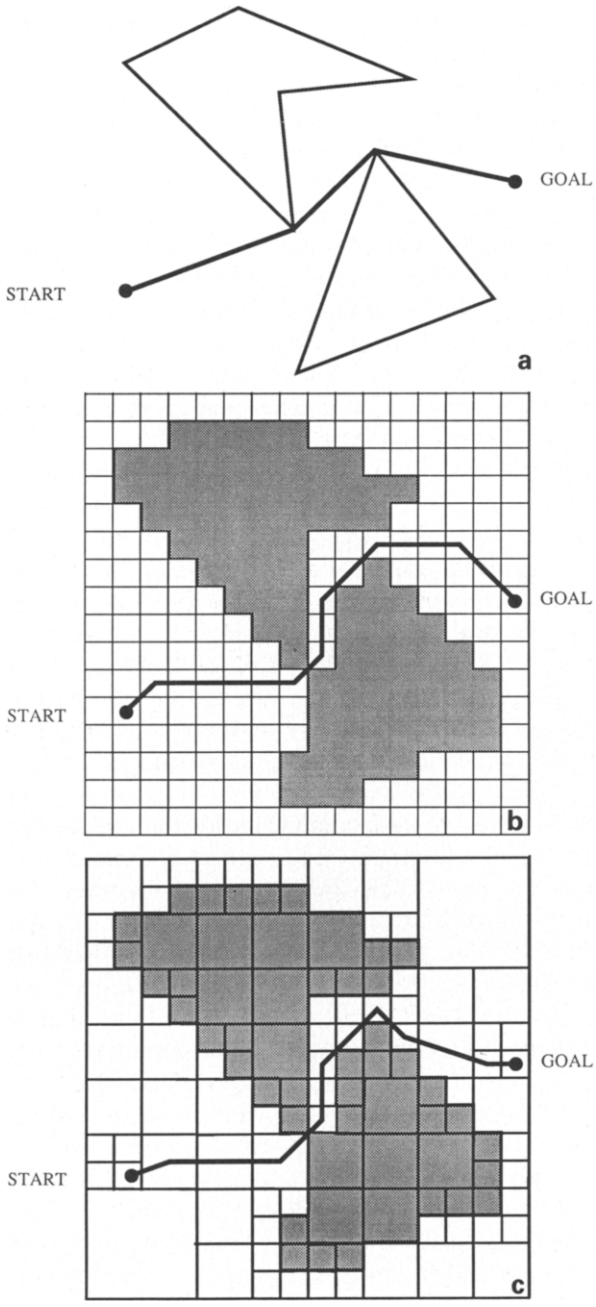


Fig. 1(a) Polygonal obstacles and a shortest path. (b) Digitized representation. (c) Quadtree representation.

function, $h(v)$, for node $v$, namely, the straight line Euclidean distance from $v$ to the goal.

Figure 1(b) shows the path from START to GOAL which results from finding the shortest path through the grid graph. Note that the path is not shortest according to the Euclidean metric. This is a common problem with the grid graph approach—it introduces a "digitization bias", a product of requiring paths to stay on the grid graph. Within the same time bounds ($O(n \log n)$), an algorithm of Keirsey and Mitchell [33] can be run which attempts to correct for this digitization bias imposed by requiring paths to stay on the grid graph. See also [57, 85, 86]. These grid-based planners do not, however, solve the exact Euclidean shortest path problem.

For maps given in the form of a list of polygonal obstacles, there are two basic approaches to finding shortest paths. A survey of these methods is given by Mitchell and Papadimitriou [55] or Mitchell [49].

The first approach is to build a *visibility graph* of the obstacle space, whose nodes are the vertices of the obstacles and whose edges connect pairs of vertices for which the segment between them does not cross the interior of any obstacle. The resulting graph can be searched using Dijkstra's algorithm or A*, where the heuristic function may again be taken to be the straight line distance from a point to the goal. The time required for searching the graph is $O(K + n \log n)$, where $K$ is the number of edges in the visibility graph. The result of the search is actually more than just the shortest path from START to GOAL; we actually get a *shortest path tree* on the set of obstacle vertices which gives a shortest path from the START (or the GOAL) to every other obstacle vertex.

The bottleneck in the worst-case complexity for the visibility graph approach is in the construction of the graph. We need to determine for every pair of vertices whether or not the vertices are visible to each other. There are $O(n^2)$ such pairs. A trivial $O(n^3)$ algorithm simply checks every pair of vertices against every obstacle edge. With a little more care, $O(n^2 \log n)$ suffices to compute the visibility graph [38, 49, 79]. The basic idea is to sort (angularly) the set of all vertices about each vertex and then to do an angular sweep about each vertex, keeping track of the closest obstacle boundary. Using a trick from duality theory that maps the vertices to lines, Lee and Ching [39] showed that the $n$ sorts can be done in total time $O(n^2)$ rather than $O(n^2 \log n)$. This fact allowed Welzl [87] and Asano et al. [3] to achieve an $O(n^2)$ algorithm for constructing visibility graphs. This running time is not likely to be improved for the worst case since there are visibility graphs of quadratic size.

However, the most recent development in the analysis of visibility graphs is the algorithm of Ghosh and Mount [23] which computes the visibility graph of a set of $n$ disjoint line segments in time $O(K + n \log n)$, where $K$ is the number of edges in the visibility graph. This *output-sensitive* algorithm takes advantage of the fact that $K$ is not always of size $\Theta(n^2)$, but rather can be as small as

$\Theta(n)$. So, when the visibility graph is sparse (say, $O(n \log n)$ edges), the algorithm is very fast, building the visibility graph in time $O(n \log n)$. Thus, when $K = O(n \log n)$, Dijkstra's algorithm (or A*) requires worst-case complexity $O(K + n \log n) = O(n \log n)$, so the shortest path can be found in time $O(n \log n)$.

A second technique for finding shortest paths is that of building a *shortest path map*. A shortest path map is a subdivision of the plane into regions each of which is the locus of all goal points whose shortest paths from the START have the same topology (i.e., pass through the same sequence of obstacle vertices.) With such a subdivision, the shortest path from START to GOAL is obtained simply by locating the GOAL point in the subdivision (which takes time $O(\log n)$) and then backtracking an optimal path. See Lee and Preparata [40] or Mitchell [49] for precise definitions and examples. Reif and Storer [69] have given an algorithm to compute the shortest path map in time $O(kn + n \log n)$, where $k$ is the number of obstacles. Note that this is an improvement over $O(n^2)$ when the number of obstacles is small in comparison with the number of obstacle vertices. Their algorithm simulates the propagation of a "wavefront" from the START, in much the same way that expansion operates in the Dijkstra or A* algorithms. Because of its similarity to the discrete Dijkstra algorithm, this technique has come to be called the "continuous Dijkstra" paradigm.

The continuous Dijkstra paradigm, in the form of a plane sweep algorithm, has lead to $O(n \log n)$ algorithms in a few special cases [40, 49, 79]. Basically, if the obstacles are known to have disjoint projections onto some line (e.g., if the obstacles are vertical line segments), then the shortest path will always be monotone with respect to that line (except, perhaps, at the ends of the path). This monotonicity allows one to use plane sweep algorithms for the iterative construction of the shortest path map.

In the simple case of finding shortest paths required to stay inside a single simple polygonal room with $n$ sides (without other obstacles), the shortest path can be found in linear ($O(n)$) time once a triangulation of the polygon is known [25, 40]. (Triangulation of a simple polygon is now known to require $o(n \log n)$ time, as Tarjan and van Wyk [84] have provided an $O(n \log \log n)$ algorithm. It is suspected that triangulation can be done in linear time, but this is another outstanding open problem in computational geometry.) Furthermore, a triangulated simple polygon can be processed in linear time so that in time $O(\log n)$ one can find the length of the shortest path from any given START to any given GOAL [26]. Furthermore, in time proportional to the number of bends in the optimal path, one can output the actual path from START to GOAL.

Lower bounds are known for the shortest obstacle-free path problem. In the worst case, computing the shortest path among $k$ obstacles with a total of $n$ vertices requires $\Omega(n + k \log k)$ time. One can use a shortest path algorithm to

find convex hulls, which in turn can be used to sort integers, for which the lower bound is well known (see [65]).

The big open problem in this field is to determine whether or not the shortest path among $n$ disjoint line segments can be computed in optimal time $\Theta(n \log n)$, and more generally if shortest paths among $k$ simple polygons can be found in time $\Theta(n + k \log k)$. We should note that for some metrics other than Euclidean, such as the $L_1$ ("Manhattan") metric and "fixed orientation" metrics [88], optimal or near-optimal shortest path algorithms are known (see [15, 17, 50]).

## 3. Shortest Paths on a Surface

A natural generalization of the obstacle avoidance problem in the plane is to consider the case of finding shortest paths for a point which is constrained to move on a nonplanar (possibly nonconvex) surface. To see that this is a generalization of the above problem, consider the surface which results from making each obstacle into a very tall cylinder whose base is in the plane of motion. Then, one can show that the shortest path on such a (nonconvex) surface will stay in the plane and follow the shortest obstacle avoidance path. Note that the problem we wish to solve is a calculus of variations problem: Find the shortest *geodesic* path between two points on a surface. Since the surface is modeled in a discrete manner, we refer to this problem as the *discrete geodesic problem*.

Here, we assume that our vehicle is not able to fly, but is able to travel over all types of terrain with uniform efficiency. We can go up hills, down hills, and even perhaps on the underside of an overhang. (Our tires are "magnetic" and the ground has a very high iron content.) Such is the problem faced by the crawling insect in Fig. 2. He desires to get to the cheese in the shortest possible path, but he does not have wings to be able to fly there. Along what route should he crawl?

First, let us mention something about the representation of the problem. What is the map in this case? It is simply some representation of a surface in three dimensions. Formally, the surface might be represented as a polyhedral surface, giving faces, edges, and vertices, along with all of the usual adjacency relationships that allow one to give the faces on each side of any edge and to "walk around" a vertex while enumerating adjacent faces. (The winged-edge data structure of Baumgart [6] or the similar quad-edge data structure of Guibas and Stolfi [27] would work nicely here.) Another possible surface representation that is often used in practice is that of contour lines. We might be given a set of iso-elevation curves, as is usually the case with geological maps of terrain. A third common representation is that of an *elevation array*. In this case, we are simply given a two-dimensional array of numbers which represent the altitude at each grid point. Digital terrain data bases of the form

compiled by the Defense Mapping Agency fall into this category. (Typically, pixels are of size 5, 12.5, or 100 meters.) These are just some of the methods of specifying a discrete approximation of a surface. Note that the last two representations require that the surface be the image of a single-valued function of the plane (i.e., there are no "overhangs"), an assumption that generally holds for most outdoor terrain.

The case of a surface represented with an elevation array can be solved easily if one makes the approximation that paths are constrained to follow the corresponding grid graph. We just connect each pixel to each of its eight neighbors, assigning a length measured by the three-dimensional Euclidean distance, and then search the graph for a shortest path using Dijkstra or an A* algorithm. This requires $O(n \log n)$ time in the worst case, where $n$ is the number of array elements (pixels).

Research in the field of algorithmic motion planning has favored the representation of a surface in terms of a polyhedron. The general problem of finding shortest paths for a point in three dimensions amidst polyhedral obstacles has been of interest for some time [1, 2, 14, 20, 63, 69, 79]. The general problem seems very hard to solve, as Canny and Reif [10] have shown the problem to be NP-hard, and the best known exact algorithm is the singly exponential algorithm of Reif and Storer [69]. Papadimitriou [63] and Clarkson [14] have solved the problem approximately with fully polynomial approximation schemes (meaning that the running time is polynomial in $n$, $K$, and $1/\varepsilon$,

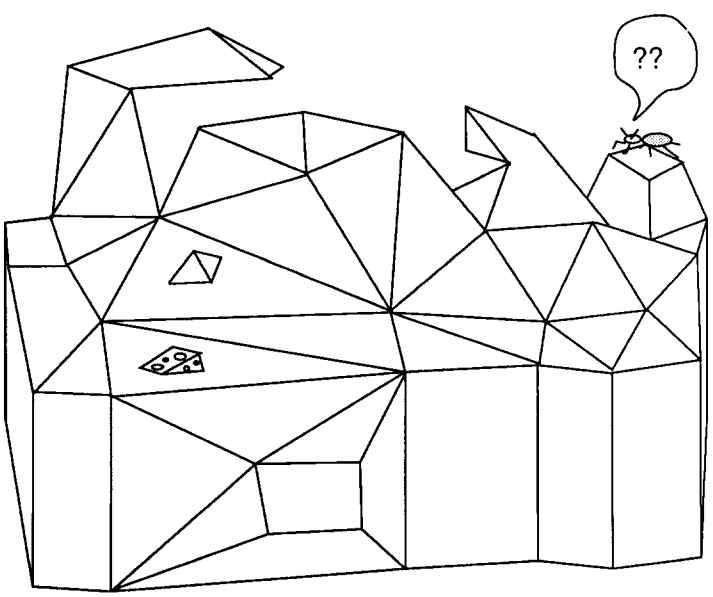


Fig. 2. The problem of navigating on a surface.

where $K$ is the number of bits to represent the problem instance). The complexity of their algorithms is $O(n^3K^2/\varepsilon)$. The special case, though, of our bug crawling on the surface of a single polyhedron is interesting because it admits a polynomial-time algorithm which is exact.

Franklin and Akman [20] and Akman [1, 2] have given exponential (worst-case) algorithms for the shortest path problem on a surface, and have shown that their algorithms perform reasonably well in practice. The first polynomial-time algorithm was an $O(n^3 \log n)$ algorithm of Sharir and Schorr [79] which worked only for the case of convex polyhedra (where $n$ is the number of edges of the polyhedron). Most interesting terrain is not convex, though, since it will usually have many mountains and valleys. An algorithm which works for nonconvex surfaces was devised by O'Rourke et al. [62], and its complexity is $O(n^5)$. Both of these running times were improved by the algorithm of Mitchell et al. [54], which runs in time $O(n^2 \log n)$. The algorithm actually constructs, for a fixed start point, a subdivision (a *shortest path map*) of the surface such that after the $O(n^2 \log n)$ preprocessing required to build the subdivision, the distance to any particular destination can be found by locating the goal in the subdivision (a task which is well known to take $O(\log n)$ time; see, for example, [34]). Listing of a path can be done in time proportional to the number of turns in the optimal path.

Let us make a few simple observations that were cricital to all of the algorithms mentioned above. We can state a few known facts about how optimal paths must behave:

(1) Optimal paths are simple (not self-intersecting), and they can pass through any one face at most once.

(2) Optimal paths "unfold" into straight lines. This is an observation that any high school geometry student has made if he has tried to solve the problem of finding the shortest path between two points on the surface of a box: Simply "cut open" the box and flatten it in such a way that the start and goal can be connected by a straight line segment that stays on the flattened cardboard. One such unfolding is guaranteed to produce a shortest path. An optimal path which passes through an edge of the polyhedron must obey the *local optimality criterion* that it unfolds to give a straight line. More generally, for any type of surface there is a similar statement of the principle of optimality, and this produces a characterization of the *geodesics* of that type of surface. (Recall that the geodesics of a sphere are arcs of great circles; the geodesics of a polyhedral surface are characterized by the above unfolding property.) Of course, the question that must be answered by an algorithm is which of the (possibly exponentially many) unfoldings yields a shortest path?

(3) On a convex surface, optimal paths will not go through vertices (except perhaps at their endpoints), while on a nonconvex surface, an optimal path may pass through a vertex only if the angle it makes in so doing is greater than $\pi$ (see [54] for a more precise statement).

Obviously, any algorithm which attempts to solve a shortest path problem should exploit as much of the structure of the paths as possible. It is also sometimes possible to exploit special structure when designing good heuristics. For example, to solve a shortest path problem on a given surface, we could use the simple heuristic which connects the start with the goal along the path "as the crow flies" (that is, the intersection of the surface with the vertical plane through the start and goal), and then iteratively improves the path by applying the local optimality criterion of (1) above. This will result in a *locally* optimal path which will frequently be very good in practice (while it is certainly not guaranteed to be *globally* optimal). Perhaps a planner which combines some "intelligence" with this heuristic would produce reasonably good paths. For example, one might have knowledge that there is only one feasible way through a mountain range, say through a specific gorge, and we would then want to route our initial guess through this passage, so that the locally optimal path produced will be reasonable. Further experimental and theoretical research is needed on these problems.

## 4. Weighted Regions

We now consider a more general terrain navigation problem. Assume that our map represents a set of *regions* in the plane, each of which has an associated *weight*, or cost, $\alpha$. The weight of a region specifies the "cost per unit distance" of a vehicle (considered to be a point) traveling in that region. Our objective is then to find a path from START to GOAL in the plane which minimizes total cost according to this weighted Euclidean metric. For example, the weights may represent the reciprocal of the maximum speed in each region, in which case minimizing the weighted Euclidean length of a path is simply minimizing the length of time it takes the vehicle to execute the path.

This problem has been termed the *weighted region problem*. Note that it is a generalization of the obstacle avoidance problem in the plane, for the obstacle avoidance problem is simply the weighted region problem in which the weights are either 1 or $+\infty$ depending on whether a region is "free space" or obstacle, respectively. Thus, an important theoretical reason for considering the weighted region problem is that it is a natural generalization of the well-studied obstacle avoidance problem.

An example of a typical map is illustrated in Fig. 3. Basically, the ground surface is subdivided into uniform regions, with a label (or some traversability index) attached to each region which gives information about how fast one can move in that region (or how costly it is to do so). Presumably, the robot can move through different types of terrain at different speeds. Speeds may also depend on other factors such as time of day, precipitation, or the location of other vehicles; however, we assume that a given problem instance has fixed weights. In military applications, there may be regions which correspond to high threat risk, perhaps because the enemy has good visibility of you when

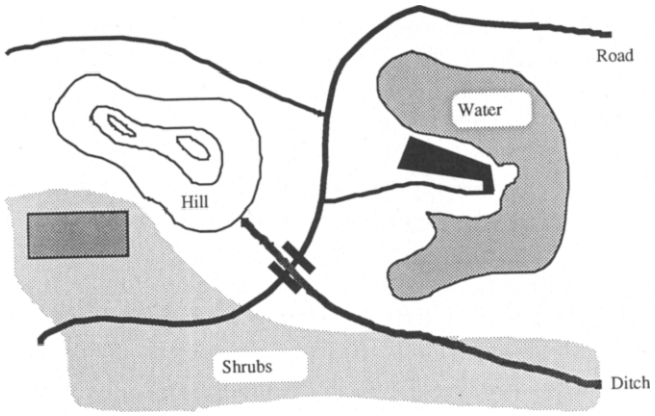


Fig. 3. Map for terrain navigation.

you are in these regions. Costs can be assigned to traveling in these risky regions as well.

We will discuss two basic types of map representations: regular tesselations (e.g., grids of pixels, or quadtrees), and straight line planar subdivisions.

Representing terrain in the form of a regular grid of pixels is natural and simple. Figure 4 shows the map of Fig. 3 in digitized form, on a grid of size $26 \times 40$ (1040 pixels). Frequently, terrain data is given in the form of a set of arrays, with each array giving information about some aspect of the terrain (e.g., ground cover, land usage, hydrography data, man-made features, and traversability indices). This, for example, is the form in which the Defense

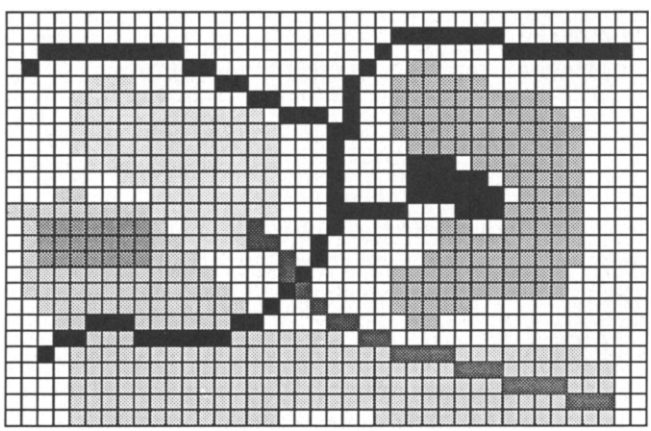

Fig. 4. Digitized terrain map.

Mapping Agency terrain data is supplied. Pixels are usually squares 5, 12.5, or 100 meters on a side. From this type of data, we can specify the map by taking the superposition of all the data arrays to form one composite map (a "terrain array"), whose regions of equal attributes will have descriptions such as "brush-covered, drainage plain", or "forested land, sandy soil, small boulders". Each composite region may be assigned a weight representing the cost of motion in that type of terrain. Certain types of terrain features have priority over others. For example, if the "roads array" tells us there is a road surface occupying pixel $(i, j)$ while the ground cover array describes the pixel as "grassland", then the composite region attribute at $(i, j)$ should say that there is a road there.

If the terrain map is given as a grid of weighted pixels, then a straightforward solution to the weighted region problem is to search the corresponding grid graph for shortest paths [29, 33, 66]. This is simply a generalization of the approach described earlier for the case of digitized obstacle maps. The grid graph will have either 4- or 8-connectivity depending on whether or not we consider diagonal neighbors of pixels. Costs may be assigned in the natural way to arcs connecting adjacent pixels of the grid, taking into account the weights of the two pixels by, say, taking the cost of the arc between them to be their average weight. Then, Dijkstra's algorithm or A* may be used to compute minimum-cost paths in the grid graph. For a grid with $n$ pixels, this results in an algorithm that runs in worst-case time $O(n \log n)$. This approach is particularly appealing in cases in which the data is given to us in the form of feature arrays.

One problem with the grid graph approach is that it may require extremely fine grids to capture the content of a relatively simple piece of terrain. Frequently, there may be just a couple of large uniform regions, perhaps with a road running through them, and the pixel representation of the map requires, say, a $512 \times 512$ array. The shortest path may indeed be obvious (such as a simple straight line segment), but the algorithm must still search and expand thousands of pixels during its execution (although hierarchical algorithms can avoid much of the expansion). Another problem with the grid graph approach is that it creates a *digitization bias* because of the metrication error imposed by confining movements to 4 or 8 orientations. See Keirsey and Mitchell [33] for a more detailed discussion of digitization bias and the various possible remedies for it. See Quek et al. [66] for a technique that uses grid graphs of multiple resolutions to design a hierarchical algorithm.

If, instead of being a regular tesselation, the map is modeled as a straight line planar subdivision then the regions are just simple polygons. Figure 5 shows the map of Fig. 3 drawn as a straight line planar subdivision with 85 edges. An appropriate data structure would be either a winged-edge [6] or a quad-edge [27] data structure. Roads and other linear features (such as ditches, fences, streams, etc.) can be modeled as very skinny polygonal regions or

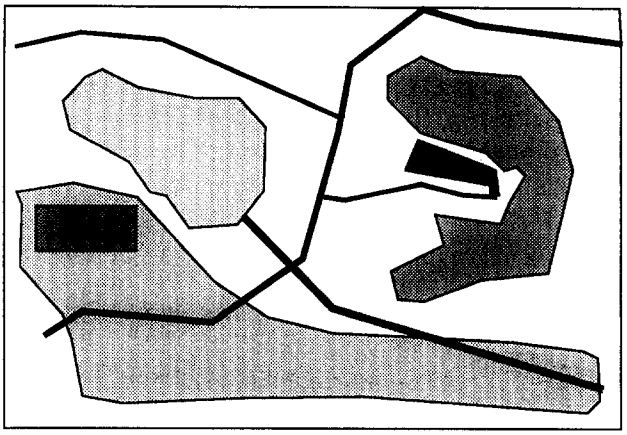

Fig. 5. Polygonal subdivision map.

approximated as sets of line segments. In the planar subdivision, then, we allow edges to be assigned a weight which may be different from the weights of the faces on either side of it. If an edge is a line segment of a road, then, presumably, its weight will be less than the weights on either side of it (assuming that it is cheaper to travel on roads than off roads). If an edge is a line segment of a ditch or a fence, then it may be assigned a fixed cost of crossing it. The fixed cost would be $+\infty$ if the edge cannot be crossed.

The representation of regions as polygonal patches and of roads as linear features has been used in the work on the DARPA Autonomous Land Vehicle project. One advantage of encoding the map information originally in the form of polygonal patches is that it can save greatly on the storage costs, while increasing the resolution of the data. The centerline of a road need not be specified only to the resolution of a pixel (say 12.5 meters), but can be specified to any desired degree of accuracy and can be specified to a higher resolution than other less critical features of the terrain.

In order to put the terrain array into the representation of a polygonal subdivision, we must fit polygonal boundaries about the sets of pixels with uniform attributes. At the highest resolution, each pixel can be considered to be a polygonal patch, namely, a square; however, this extreme will usually result in many more regions than desired, and it does not reflect the fact that the original data was only approximate in the first place. Usually, we will also want to represent roads, fences, and power lines as linear features by fitting a piecewise-linear path to them.

Assume now that we are given a planar subdivision with weights. One approach to solving the weighted region problem is to build a "region graph" in which nodes correspond to regions and arcs correspond to boundaries between adjacent regions. Assume that the subdivision is fine enough that all

regions are convex. (We could, for example, make the subdivision a triangulation each of the simple polygonal regions; however, we may want to subdivide in such a way that each convex region is as close to being circular as possible.) Then we can think of placing a node at the "center" (e.g., center of mass) of each region. Two nodes are joined by an edge if the corresponding regions are adjacent. We then assign costs to arcs according to the weighted distance between adjacent nodes. (An alternative graph can be constructed by placing nodes at the midpoints of region edges and linking two nodes if the corresponding edges share a common region.) Note that a grid graph is a special case of a region graph in which all the regions are equal-sized squares. Searching this graph for shortest paths yields a "region path" from the START to the GOAL, giving a sequence of regions through which a "good" path should pass. We could then do some postprocessing (e.g., using Snell's Law of Refraction from optics, a local optimality criterion that we will mention below) to make the path locally optimal at region boundaries. The problem with this approach is that it can produce paths that are far from being optimal, for the optimal path need not have any relationship to the shortest region-path. In practice, however, it promises to be a very useful technique, particularly in cases in which the data is not very accurate or the assignment of weights is somewhat subjective. (Can one really say that grassland deserves a weight of 5 while brushland deserves a weight of 15? It seems likely that weights will be highly subjective.)

Thus, the region graph approach does not guarantee that the resulting path is close to being optimal. Another approach given by Mitchell and Papadimitriou [56] yields a polynomial-time algorithm for computing $\varepsilon$-optimal paths. The length of the produced path is guaranteed to be within a factor of $1 + \varepsilon$ of the length of an optimal path. The complexity of the algorithm is $O(n^7 L)$, where $L$ is the number of bits needed to represent the problem data and $\varepsilon$. The algorithm is a generalization of the algorithm for the discrete geodesic problem given by Mitchell et al. [54], and it too uses the continuous Dijkstra paradigm.

Let us be more precise about the problem solved in [56]. We are given a (straight line) planar subdivision, $\mathcal{S}$, specified by a set of faces (regions), edges, and vertices, with each edge occurring in two faces and two faces intersecting either at a common edge, a vertex, or not at all. We consider faces to be *closed* polygons (they include their boundaries) and edges to be *closed* line segments (they include their endpoints, which are vertices). We are also given an initial point, START. Without loss of generality, we assume that all faces are triangles and that $s$ and $t$ are vertices of the triangulation. Assume that $\mathcal{S}$ has $n$ edges (and hence $O(n)$ triangular faces and $O(n)$ vertices). Our complexity measures will be written in terms of $n$.

Each face $f$ has associated with it a weight $\alpha_f \in [0, +\infty]$ which specifies the cost per unit distance of traveling on face $f$. Similarly, each edge $e$ has associated with it a weight $\alpha_e \in [0, +\infty]$. The weighted distance between any

two points $x$ and $y$ on edge $e$ is simply the product $\alpha_e|xy|$, where $|xy|$ is the usual Euclidean distance between $x$ and $y$. Likewise, the weighted distance between any two points $x$ and $y$ on face $f$ (but not both on the same edge of $f$) is the product $\alpha_f|xy|$. The weighted length of a path through the subdivision is then the sum of the weighted lengths of its subpaths through each face.

We are asked to find the minimal length (in the weighted sense) path from START to some goal point. The algorithm of [56] actually solves the *query* form of the weighted region problem: Build a structure which allows one to compute a shortest path (in the weighted Euclidean metric) from START to *any* query point $t$. It turns out that this single-source version of the problem is no harder (in worst-case asymptotic time) than the problem with a given fixed goal point.

As with the discrete geodesic problem, the algorithm for the weighted region problem exploits certain facts about the local behavior of optimal paths:

(1) Shortest paths are piecewise-linear, bending only at points where they cross an edge or pass through a vertex. This follows from the fact that each region is assumed to have a *uniform* weight. (The case of *non*uniform weighted regions gives rise to other types of curves and is an interesting area for further research.)

(2) When an optimal path passes through an edge between regions, it does so while obeying Snell's Law of Refraction at that edge. This is the local optimality criterion which is analogous to the fact that shortest paths on the surface of a polyhedron unfold to be straight lines. It is a simple property of locally optimal paths through weighted regions which has been observed by many researchers (Lyusternik [43], Mitchell and Papadimitriou [55, 56], Richbourg [70], Richbourg et al. [71, 72], Rowe and Richbourg [74], Smith [80]).

**Snell's Law of Refraction.** *The path of a light ray passing through a boundary $e$ between regions $f$ and $f'$ with indices of refraction $\alpha$ and $\beta$ obeys the relationship that $\alpha \sin\theta = \beta \sin\phi$, where $\theta$ and $\phi$ are the angles of incidence and refraction (respectively).*

The *angle of incidence*, $\theta$, is defined to be the counterclockwise angle between the incoming ray and the normal to the region boundary. The *angle of refraction*, $\phi$, is defined as the angle between the outgoing ray and the normal. Refer to Fig. 6.

The fact that light obeys Snell's Law comes from the fact that light seeks the path of minimum time (this is *Fermat's Principle*). The index of refraction for a region is proportional to the speed at which light can travel through that region. Hence, the shortest paths in our weighted region problem must also obey Snell's Law. This can be shown formally [49, 56].

(3) The angle of incidence of an optimal path at a boundary that it crosses

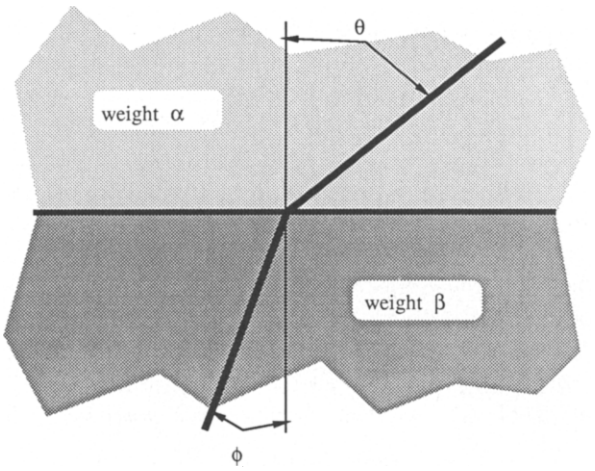


Fig. 6. Light ray crossing a boundary.

will not be greater (in absolute value) than the *critical angle* defined at that boundary. This fact comes from examining the relationship specified in Snell's Law and making sure it is well-defined. Without loss of generality, assume that $\alpha_f > \alpha_{f'} > 0$. Then, we must assure that

$$-1 \leq \sin \theta_{f'} = \frac{\alpha_f}{\alpha_{f'}} \sin \theta_f \leq 1 .$$

The angle, $\theta_c(e) = \theta_c(f, f')$, at which

$$\frac{\alpha_f}{\alpha_{f'}} \sin[\theta_c(f, f')] = 1$$

is called the *critical angle* defined by edge $e$. If $\alpha_f > 0$ and $\alpha_{f'} \geq 0$, then the critical angle is defined and is given by $\theta_c(f, f') = \sin^{-1}(\alpha_{f'}/\alpha_f)$, *provided* that $\alpha_f \geq \alpha_{f'}$. A ray of light which strikes $e$ at the critical angle will (theoretically) travel along $e$ rather than enter into the interior of $f'$. We can think of it as if the light ray wishes to be "just inside" the "cheap" region on the other side of $e$.

(4) Optimal paths will not reflect from a boundary, except when they are "critically reflected" as shown in Fig. 7: A path is incident (from a face $f$ of weight $\beta$) on edge $e$ (of weight $\gamma < \min\{\alpha, \beta\}$) at the critical angle $\phi = \theta_c$ at some point $y \in \text{int}(e)$, it then travels along edge $e$ for some positive distance, and then exits edge $e$ *back into face* $f$ at some point $y' \in \text{int}(e)$, leaving the edge at an angle $\phi = \theta_c$. We then say that the path is *critically reflected* by edge $e$ and that segment $\overline{yy'}$ is a *critical segment* of $p$ along $e$. The phenomenon of critical

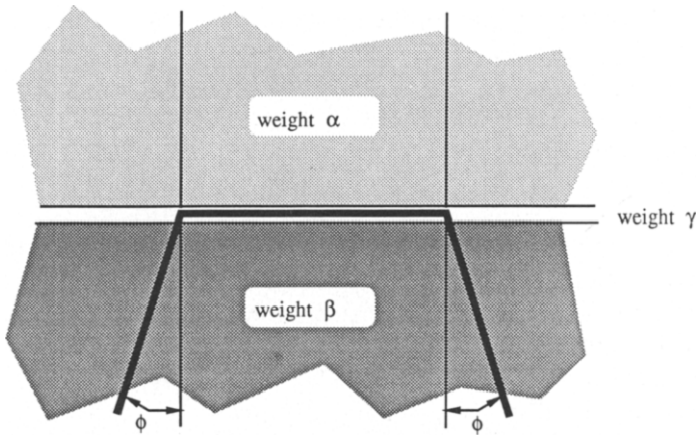


Fig. 7. An optimal path which is critically reflected.

reflection is not just an unlikely event of degeneracy. It is commonly the case that the shortest path between two points consists of a single critical reflection along an edge.

(5) If $\alpha_e > \min\{\alpha_f, \alpha_{f'}\}$, then an optimal path will never travel along edge $e$, since it would be better to travel "just inside" either region $f$ or region $f'$ (whichever is cheaper).

(6) An optimal path can cross a "road" (that is, an edge whose cost is less than that of the regions on either side of it) in one of two ways: either by obeying Snell's Law while passing through the boundary, or by "hitching a ride" along the edge, as illustrated in Fig. 8. The path hits edge $e$ at the *incoming* critical angle $\phi = \theta_c(f, e) = \sin^{-1}(\alpha_e/\alpha_f)$, then travels along edge $e$ (of

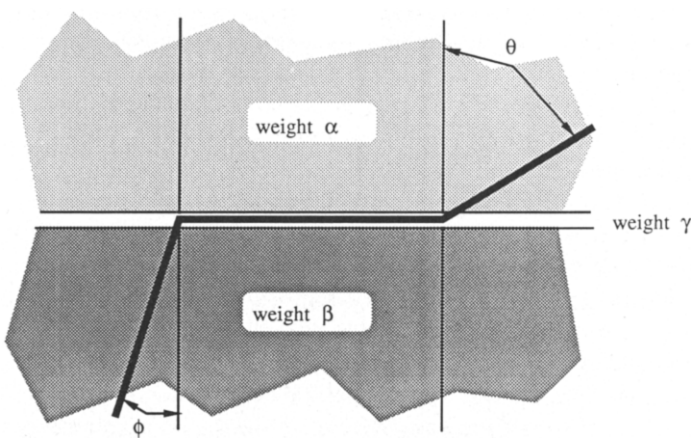


Fig. 8. Shortest path "hitching a ride" on a road.

weight $\alpha_e$) for some distance, then leaves edge $e$ into $f'$ at the *outgoing* critical angle $\theta = \theta_c(f', e) = \sin^{-1}(\alpha_e/\alpha_{f'})$.

(7) On an optimal path, between any critical point of exit and the next critical point of entry, there must be a vertex of $\mathcal{S}$. This fact turns out to be very important in the proofs of the complexity of the algorithm of Mitchell and Papadimitriou [56].

The local optimality criteria outlined above are sufficiently strong that they uniquely specify a locally optimal path which passes through a given sequence of edges. (This follows from the global convexity of the objective function when written in terms of the coordinates at which the path crosses each edge.) The problem remains, however, of how to select the proper sequence of edges for the optimal path.

The continuous Dijkstra algorithm of Mitchell and Papadimitriou [56] simulates propagation of a wavefront from START. Each time the wavefront first hits a region boundary, passes through a vertex, or otherwise changes its description (that is, the list of regions through which it passes), we record the effect of the event in an appropriate data structure. When the wave has propagated throughout the surface, we have completed the construction of a shortest path map. For the application to the weighted region problem, it can be shown that the worst-case number of *event* points is $O(n^3)$, thereby establishing a polynomial-time bound.

Processing each event involves calling certain functions. One such function locates a point in an "access channel" between representative optimal paths that have already been established. This is easy and can be done by binary search in $O(\log n)$ time. However, we must also do $O(n^3)$ operations of the form "Find the refraction path from a given point $r$ that passes through a given sequence of edges (while obeying Snell's Law) to hit a given point $x$". This is not an easy problem to solve exactly. It is handled by a numerical routine that does binary search to find a path whose length is at most $(1+\varepsilon)$ times the length of the optimal path, where $\varepsilon > 0$ is an arbitrary error tolerance value that is given in the problem specification. The search requires time which is polynomial in $n$ and $L$, the number of bits necessary to represent the problem instance (namely, $\log(nNW/\varepsilon w)$, where $N$ is the largest integer value of any vertex coordinate, $W$ is the largest finite weight, and $w$ is the smallest nonzero weight). A crude calculation shows that the time to run the numerical search is bounded by $O(n^4 L)$.

It would be great if we could calculate exactly the refraction path from point $r$ to point $x$, but this problem seems to be hard. The complication is that when we write down the equations that represent the local optimality criterion (Snell's Law) at each edge, and then try to solve for the points where the path crosses edges we get $k$ quartic equations in $k$ unknowns (where $k$ is the length of the edge sequence through which the refraction path is known to pass).

Elimination among these equations yields a very high-degree polynomial (of degree exponential in $k$). Alternatively, we could apply the cylindrical decomposition technique of Collins [16] to arrive at a doubly exponential-time procedure to determine whether or not a given rational path length is achievable. This would also provide us with a technique of comparing the lengths of paths through two different edge sequences, and thus would solve the entire problem precisely in doubly exponential time (outputting the sequence of edges and vertices along the optimal path). The same technique led Sharir and Schorr [79] to a doubly exponential-time solution to the three-dimensional shortest path problem.

In practice, a very straightforward numerical approach to solving the search problem can be used. For example, one technique begins with a path from $r$ to $x$ that connects the edges in the sequence along their midpoints. We then iteratively shorten the path by applying the local optimality criterion to adjacent segments of the path. We simply pick an edge at which Snell's Law is violated, and then let the crossing point "slide" along the edge until Snell's Law is obeyed. (This is a coordinate descent method for searching for the minimum of the convex function which describes the path length.) Each iteration can be computed in constant time and results in a strict decrease in the weighted path length. Hence, this procedure will converge to the locally optimal path from $r$ to $x$, which is also the retraction path from $r$ to $x$. The problem with this approach is that we do not have good bounds on the rate of convergence or on the number of iterations necessary to guarantee that the solution is close (say, within $\varepsilon\%$) to optimal.

In practice, however, the convergence rate has been observed to be "very fast". In particular, in some recent implementation work of Karel Zikan [89], he has observed that a version of the abovementioned coordinate descent method tends to take only about 5–10 iterations to get within 1 percent of optimality for problems with up to about 20 edges. By employing certain "overrelaxation" methods, he is able to improve the convergence rate, cutting approximately in half the number of iterations needed.

## 5. Special Cases of the Weighted Region Problem

There are a few special cases of the weighted region problem which admit alternative polynomial-time algorithms and are of interest in terrain navigation. We have already mentioned the special case in which all weights are 1 or $+\infty$. This leads to the usual obstacle avoidance problem in the plane.

Consider now the special case in which all the weights on faces and edges are 0, 1, or $+\infty$. Infinite-cost regions are obstacles, cost-one regions are simply "free space" (through which we can move at some fixed speed), and cost-zero regions are "freebies" (through which motion costs nothing; we can move at

infinite speed). If there are no zero-cost regions, then the problem is simply the usual obstacle avoidance shortest path problem in the plane. When there are zero-cost regions, they behave like "islands" between which one may want to "hop" to get from the source to the destination.

As a motivation for this special case, let us first note that it may be a reasonable approximation to the case in which the terrain is divided into three categories: obstacles (or very costly regions), very low-cost regions, and regions whose costs are in between, but similar. Such would be the case for a runner (who is not a very good swimmer) trying to find a route from one point (say, on a flat island) to another point (say, in the water). There are many islands around. Some are flat and can be crossed easily on foot. Others have huge cliffs surrounding them which make them impassable. Swimming through the water is possible, but the runner would much prefer to be running across a flat island. So how can he get from one point to another in the shortest time? His path will "hop" from one flat island to another, swimming in between, and avoiding the mountainous islands.

An application of this problem which has arisen in practice is the *maximum concealment* problem. There are regions of the plane which are "visible" to an enemy threat, and these are hopefully to be avoided. There are also obstacles, which *must* be avoided. Then there are regions (e.g., between mountain ranges, or behind rocks) which are hidden from the enemy's view, and are hence extremely cheap in comparison with the visible regions. See Fig. 9 for an example in which there is a single enemy observer whose view is obstructed by tall obstacles, but who can see over the bodies of water. The tall obstacles cast "shadows", which are regions of essentially free travel. The problem is to find an obstacle-avoiding path between two points which maximizes the concealment (minimizing the length of time in the exposed regions). In the context of the weighted region problem, the model is that water and mountain ranges are infinite-weight regions, shadowed regions behind mountains are zero-cost regions (for purposes of concealment), and all other terrain has weight 1. (It makes sense that the cost of being seen by a threat should depend on the distance from the threat. Models that make this assumption are discussed by Mitchell [53], where polynomial-time algorithms are given for some special cases.)

Other applications also use the $\{0, 1, +\infty\}$ special case of the weighted region problem. It can be used to find the shortest path from one *region* to another *region* in the presence of obstacles (by letting the source and destination regions be zero-weight regions), as was independently considered by Asano et al. [4]. It can also be used to find lexicographically shortest paths through weighted regions: minimize the length of path in the most expensive region, then, subject to this being minimal, minimize the length in the next most costly region, etc. Lexicographically shortest paths may be good initial

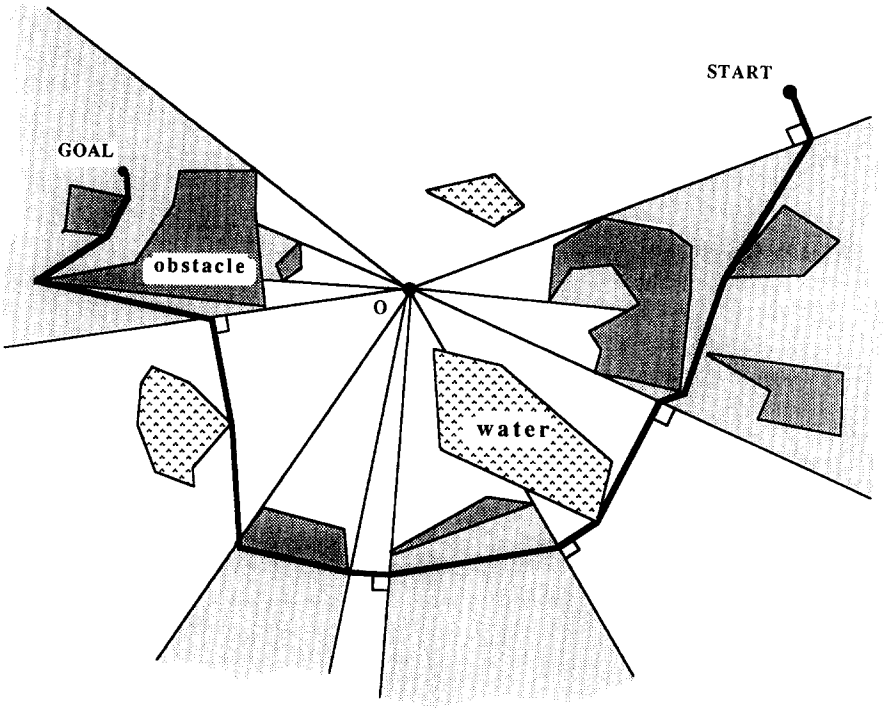


Fig. 9. Maximum concealment from a single enemy observer.

guesses for heuristic algorithms that try to find shortest paths through weighted regions. Also, there is an intimate connection between shortest path problems and *maximum flow* problems, and the algorithms for maximum flow need to find shortest paths through $\{0, 1, +\infty\}$ weighted regions. These and other applications are discussed in Mitchell [51, 52].

An interesting application discussed by Gewali et al. [22] is that of the "least-risk watchman route problem". The watchman problem asks one to find the shortest path that a watchman would have to make from his current location in order to be able to "see" all of a given space (e.g., a polygonal room, or a polygonal room with "holes"). Chin and Ntafos [12, 13] have shown that the optimum watchman route problem is NP-complete in a polygonal room with holes, but that there is a polynomial-time algorithm for polygonal rooms without holes. The *least-risk watchman route problem* asks one to find the path of minimum exposure to threats which sees the entire space. Then, for rectilinear polygons with $n$ sides, Gewali et al. [22] give an $O(k^2n^3)$ algorithm for the case of $k$ threats.

The algorithm described by Mitchell and Papadimitriou [56] for the weighted region problem does in fact solve the $\{0, 1, +\infty\}$ special case in polynomial

time. But the special case has structure which allows an alternative approach which is faster and easier to implement. The approach by Mitchell [51] and Gewali et al. [22] to this case is to build a special kind of "extended visibility graph", $VG^*$, after shrinking the zero-cost regions to single nodes. The graph $VG^*$ can be constructed in polynomial time (in fact, quadratic time), and it can then be searched for shortest paths in the usual way using Dijkstra or A*. The result is an exact polynomial-time algorithm. The basic idea is to exploit the local optimality criteria, which allow the problem to be discretized by limiting our search to the graph $VG^*$.

(1) The behavior of optimal paths with respect to the obstacles is the same as it was in the obstacle avoidance problem (see [49]). Namely, the path behaves around obstacles as if it were a "taut string", being locally tangent to obstacles and forming a convex angle when it comes in contact with an obstacle vertex.

(2) An optimal path will either enter a zero-cost region at a vertex or at a normal to the interior of an edge bounding the region. If it enters at a vertex, then it must make an angle greater than $\frac{1}{2}\pi$ with the edges incident to the vertex.

Thus, the extended visibility graph has edges of two types: those that join two vertices (of obstacles or zero-cost regions), and those that join a vertex (of either an obstacle or a zero-cost region) to an edge of a zero-cost region (in such a way as to be perpendicular to the edge). This immediately implies a quadratic bound on the size of the graph. The exact details of the construction are given in either of the abovementioned papers [22, 51]. The result is an $O(n^2)$ algorithm for shortest paths.

Another special case of interest in terrain navigation is that in which all of the terrain has the same (finite) weight, but there exists a (piecewise-linear) network of roads or other linear features (of possibly different sizes, which admit different speeds of traversal). This is the case illustrated in Fig. 10, and is simply the weighted region problem in which all face weights, $\alpha_f$, are identical, and edge weights, $\alpha_e$, are arbitrary. Then, we apply the following local optimality criterion:

(3) Shortest paths will enter a road segment either at one of its endpoints, or by hitting at the critical angle of incidence defined by the ratio of the edge's weight to that of the surrounding terrain.

This observation again leads to a discrete graph (a *critical graph*) which can be searched for shortest paths. Mitchell [51] and Gewali et al. [22] show how to construct this critical graph and search it in time $O(n^2)$, where $n$ is the number of road edges. See also [73].

A generalization of this technique is possible by putting together all of the

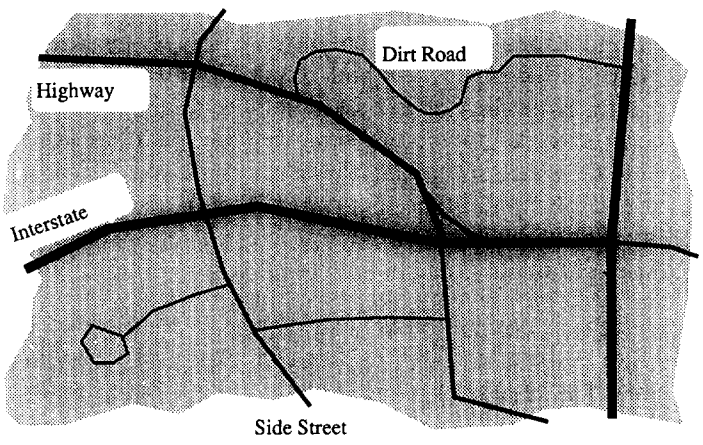


Fig. 10. A map consisting of just roads.

abovementioned local optimality criteria. Assume now that we have a weighted region problem in which the weights on faces are in the set $\{0, 1, +\infty\}$, and the weights on edges are arbitrary ($\in [0, +\infty]$). Additionally, there can be a fixed cost, $\xi_e$, associated with crossing an edge $e$. The model is that there are obstacles, zero-cost regions, and background regions (all of the same weight). Additionally, there are roads, fences, etc. of arbitrary weights. Then, the algorithm given by Mitchell [51] or Gewali et al. [22] finds shortest paths in time $O(n^2)$, again by building a critical graph.

## 6. Generalizations and Extensions

Several generalizations to the discrete geodesic and weighted region problems are possible. Some are solved by minor changes to the algorithms of Mitchell et al. [54] and Mitchell and Papadimitriou [56], while others are open research problems.

(1) First, one can generalize to the case of *multiple-source points*. This means that instead of building a shortest path map for one point, START, we consider several given point sites (called "source points"), and we compute a subdivision of the plane or the surface which tells us for each point $t$ which source point is closest to it and how to backtrace a shortest path to the closest source point. The resulting structure is then a classic *Voronoi diagram* according to the metric of interest (either shortest Euclidean paths on a surface or weighted Euclidean lengths through a weighted map).

As a further generalization along these lines, we could allow the source and goal to be *regions* rather than single points. This generality is actually already incorporated in the model of Mitchell and Papadimitriou [56] by allowing there

to be zero-cost regions. One simply makes the starting region have zero weight, and places the starting point anywhere inside; a similar accommodation is made for the goal region. The applicability of this problem is apparent: Goals are frequently specified in the form "Get me to the town square", or "Get me to Highway 95". Any point within the square or on the highway will suffice as a goal.

(2) We can generalize the cost structure from being that of a *uniform* cost per distance in a region to that of allowing a cost function (such as a linear function) in each region. The function should be specified by some fixed number of parameters, and should allow computation of geodesics. In the uniform-cost case, the geodesics within a given region are simply straight lines. In more general cases, geodesics would have to be computed by techniques of calculus of variations [43]. The algorithm must then be modified to apply the local optimality criterion (Snell's Law) to curved arcs at the boundaries of regions. If an optimal path passes through the point $y$ interior to edge $e$, then the tangent lines at $y$ to the geodesic curves in $f$ and in $f'$ must meet at $y$ such that they obey Snell's Law. Using this local optimality criterion, then, it should be possible to run an analogous algorithm to that of Mitchell and Papadimitriou [56], this time piecing together curves from region to region. The details of this generalization need to be examined further.

(3) We could allow boundaries between regions to be curved (e.g., splinegons; see [81]). Then, for the weighted region problem, Snell's Law at a point $y$ interior to a boundary curve must be applied as if the boundary at $y$ is the straight line tangent to the boundary at $y$. Again, more research is needed on this problem.

(4) The weighted region problem can be generalized to the case of a weighted polyhedral surface. Then the local optimality criterion becomes Snell's Law applied to the "unfolded problem" at each boundary. That is, for an edge $e = f \cap f'$, we must "unfold" $f'$ about $e$ so that $f$ and $f'$ are made coplanar, and then we can apply Snell's Law to points interior to $e$ in this rotated coordinate frame.

(5) We could allow the traversal of a region to be directionally dependent. This comment applies both to the (unweighted) discrete geodesic problem and to the weighted region problem. For example, it makes sense that it should be more costly to go "up" a hill than to go down it. We are currently investigating this case.

(6) An important open problem is that of incorporating other types of vehicle constraints into the shortest path problem. For example, we may want a minimum time path for a vehicle which has a bounded amount of power (implying bounded acceleration). Or it may be that the vehicle will tip over if it is on too steep of a slope. Or the vehicle may have fuel consumption constraints. Solving the exact shortest path problem in the presence of other such constraints is an important area for continuing research.

## 7. Conclusion

We have examined some of the models of shortest path problems which are relevant to route planning for an autonomous vehicle. We have discussed problems which admit exact solutions, and have discussed the currently best known algorithms. Many important issues remain to be addressed.

Obviously, there remains the question of reducing the worst-case running times of the algorithms discussed. In particular, it may be possible to improve the $O(n^7L)$ bound on the weighted region problem. It is a challenging problem to determine the *expected* running times of shortest path algorithms.

Problems involving the design of a good terrain map need to be studied more. We have discussed regular tesselations and polygonal subdivisions as representations for map data, and we have mentioned some of the benefits of each. What is probably needed is some kind of hybrid representation which can take advantage of the strengths of each, while also providing a hierarchical representation of the terrain.

If we were to compose a "wish list" of features for a map representation, we would most certainly want to include the ability to handle the following.

(1) *Discovery of new information.* Ideally, as the vehicle moves about through terrain, it will update its map with new information it "learns". It may, for example, discover that road construction is causing congestion along some stretch of highway or that a certain bridge it thought was there has been closed. It is therefore important to have a representation of a map which allows updating and annotations.

This also brings up the issue of interfacing the map-based path planner with some *sensor*-based planner that accumulates local information during the execution of a plan and must then modify the path accordingly. We have not addressed here the issue of planning routes in unknown or uncertain terrains, but we can refer the reader to a few of the myriad of papers on the subject: Goldstein et al. [24], Iyengar et al. [28], Jorgensen et al. [30], Kauffman [32], Kuan [35], Kuan et al. [36], Kuan et al. [37], Lumelsky and Stepanov [42], Meng [45, 46], Metea and Tsai [47], Mitchell [48], Mitchell et al. [57], Moravec [58], Oommen et al. [60], Rao et al. [67, 68], Rueb and Wong [75], Stern [82], Sutherland [83], and Thorpe [86].

(2) *Temporal nature of information.* Annotations might carry with them some estimated *duration* which bounds the interval of time for which the information is valid. For example, road construction is usually known to affect traffic flow only during certain times of the day, and it remains in effect only until it achieves completion.

Of course, uncertainty assures that the planner will never be able to guarantee that the routes it selects in advance are optimal. All types of map information are subject to some uncertainty. For example, roads can be closed

or become impassable, open terrain can become flooded or reforested, boundaries of lakes and rivers certainly change with the seasons and with the rainfall pattern. A truly "intelligent" planner should have information about the kinds of circumstances which potentially affect different types of terrain and be able to reason about it. If it has knowledge, for example, that there was recently a greater than average rainfall, it should adjust the probabilities it places on the traversability of a low-water bridge.

Many open questions remain. How can these issues be addressed within the framework of exact or approximate algorithms? What are the appropriate mathematical models of the path planning process? How can our knowledge of algorithmic path planners be translated into useful heuristics for complex terrain problems?

## ACKNOWLEDGMENT

This research was partially supported by a generous grant from the Hughes Aircraft Company. Part of this research was conducted while J. Mitchell was supported by a Howard Hughes Doctoral Fellowship at Stanford University and affiliated with the Hughes Artificial Intelligence Center in Calabasas, California. This research was also partially supported by a grant from the National Science Foundation (IRI-8710858).

## REFERENCES

1. Akman, V., Shortest paths avoiding polyhedral obstacles in 3-dimensional Euclidean space, Ph.D. Thesis, Computer and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY (1985).
2. Akman, V., Unobstructed shortest paths in polyhedral environments, in: G. Goos and J. Hartmanis, (Eds.), Lecture Notes in Computer Science **251** (Springer, New York, 1987).
3. Asano, T., Asano, T., Guibas, L., Hershberger, J. and Imai, H., Visibility of disjoint polygons, *Algorithmica* **1** (1986) 49–63.
4. Asano, T., Asano, T. and Imai, H., Shortest paths between two simple polygons, *Inf. Process. Lett.* **24** (1987) 285–288.
5. Baker, B., Shortest paths with unit clearance among polygonal obstacles, in: *Proceedings SIAM Conference on Geometric Modeling and Robotics*, Albany, NY (1985).
6. Baumgart, B.G., A polyhedron representation for computer vision, in: *AFIPS Conference Proceedings* **44**, *National Computer Conference* (1975) 589–596.
7. Brooks, R.A., Solving the find-path problem by good representation of free space, *IEEE Trans. Syst. Man Cybern.* **13** (3) (1983) 190–197.
8. Brooks, R.A. and Lozano-Pérez, T., A subdivision algorithm in the configuration space for find path with rotation, *IEEE Trans. Syst. Man Cybern.* **15** (2) (1985) 224–233.
9. Canny, J.F., The complexity of robot motion planning, Ph.D. Thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA (1987).
10. Canny, J. and Reif, J., New lower bound techniques for robot motion planning problems, in: *Proceedings 28th Annual IEEE Symposium on Foundations of Computer Science*, Los Angeles, CA (1987) 49–60.
11. Chew, L.P., Planning the shortest path for a disk in $O(n^2 \log n)$ time, in: *Proceedings First Annual ACM Conference on Computational Geometry*, Baltimore, MD (1985) 214–220.
12. Chin W.P. and Ntafos, S., Optimum watchman routes, in: *Proceedings 2nd ACM Symposium on Computational Geometry*, Yorktown Heights, NY (1986) 24–33.

13. Chin, W.P. and Ntafos, S., Watchman routes in simple polygons, Tech. Rept., Computer Science Department, University of Texas at Dallas, TX (1987).
14. Clarkson, K., Approximation algorithms for shortest path motion planning, in: *Proceedings 19th Annual ACM Symposium on Theory of Computing*, New York (1987) 56–65.
15. Clarkson, K., Kapoor, S. and Vaidya, P., Rectilinear shortest paths through polygonal obstacles in $O(n \log^2 n)$ time, in: *Proceedings Third Annual ACM Conference on Computational Geometry*, Waterloo, Ont. (1987) 251–257.
16. Collins, G.E., Quantifier elimination for real closed fields by cylindric algebraic decomposition, in: *Proceedings Second GI Conference on Automata Theory and Formal Languages*, Lecture Notes in Computer Science **33** (Springer, New York, 1975) 134–183.
17. de Rezende, P.J., Lee, D.T. and Wu, Y.F., Rectilinear shortest paths with rectangular barriers, in: *Proceedings First ACM Conference on Computational Geometry*, Baltimore, MD (1985) 204–213.
18. Dijkstra, E.W., A note on two problems in connection with graphs, *Numer. Math.* **1** (1959) 269–271.
19. H. Edelsbrunner, *Algorithms in Combinatorial Geometry* (Springer, Heidelberg, F.R.G., 1987).
20. Franklin, W.R. and Akman, V., Shortest paths between source and goal points located on/around a convex polyhedron, in: *Proceedings 22nd Allerton Conference on Communication, Control and Computing* (1984) 103–112.
21. Fredman, M. and Tarjan, R., Fibonacci heaps and their uses in improved network optimization algorithms, in: *Proceedings 25th Annual IEEE Symposium on Foundations of Computer Science* (1984) 338–346.
22. Gewali, L., Meng, A., Mitchell, J.S.B. and Ntafos, S., Path planning in $0/1/\infty$ weighted regions with applications, Extended Abstract, in: *Proceedings Fourth Annual ACM Conference on Computational Geometry*, Urbana-Champaign, IL (1988) 266–278.
23. Ghosh, S.K. and Mount, D.M., An output sensitive algorithm for computing visibility graphs, in: *Proceedings 28th Annual IEEE Symposium on Foundations of Computer Science*, Los Angeles, CA (1987) 11–19.
24. Goldstein, M., Pin, F.G., de Saussure, G. and Weisbin, C.R., 3-D world modeling with updating capability based on combinatorial geometry, Tech. Rept. CESAR-87/02, Oak Ridge National Laboratory, Oak Ridge, TN (1987).
25. Guibas, L., Hershberger, J., Leven, D., Sharir, M. and Tarjan, R., Linear time algorithms for visibility and shortest path problems inside simple polygons, *Algorithmica* **2** (1987) 309–233.
26. Guibas, L.J. and Hershberger, J., Optimal shortest path queries in a simple polygon, in: *Proceedings Third Annual ACM Conference on Computational Geometry*, Waterloo, Ont. (1987) 50–63.
27. Guibas, L. and Stolfi, J., Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams, *ACM Trans. Graph.* **4** (1985) 74–123.
28. Iyengar, S.S., Jorgensen, C.C., Rao, S.V.N. and Weisbin, C., Robot navigation algorithms using learned spatial graphs, *Robotica* **4** (1986) 93–100.
29. Jones, S.T., Solving problems involving variable terrain, Part 1: A general algorithm, *BYTE* **5** (2) (1980).
30. Jorgensen, C., Hamel, W. and Weisbin, C., Autonomous robot navigation, *BYTE* **11** (1) (1986) 223–235.
31. Kambhampati, S. and Davis, L.S., Multiresolution path planning for mobile robots, *IEEE J. Rob. Autom.* **2** (3) (1986) 135–145.
32. Kauffman, S., An algorithmic approach to intelligent robot mobility, *Rob. Age* **5** (3) (1983) 38–47.
33. Keirsey, D.M. and Mitchell, J.S.B., Planning strategic paths through variable terrain data, *Proc. SPIE Appl. Artif. Intell.* **485** (1984).

34. Kirkpatrick, D.G., Optimal search in planar subdivisions, *SIAM J. Comput.* **12** (1983) 28–35.
35. Kuan, D.T., Terrain map knowledge representation for spatial planning, in: *Proceedings First IEEE Conference of Artificial Intelligence Applications*, Denver, CO (1984) 578–584.
36. Kuan, D.T., Brooks, R.A., Zamiska, J.C. and Das, M., Automatic path planning for a mobile robot using a mixed representation of free space, in: *Proceedings First IEEE Conference of Artificial Intelligence Applications*, Denver, CO (1984) 70–74.
37. Kuan, D.T., Zamiska, J.C. and Brooks, R.A., Natural decomposition of free space for path planning, in: *Proceedings IEEE International Conference on Robotics and Automation*, St. Louis, MO (1985) 168–173.
38. Lee, D.T., Proximity and reachability in the plane, Ph.D. Thesis, Tech. Rept. ACT-12, Coordinated Science Laboratory, University of Illinois, Urbana, IL (1978).
39. Lee, D.T. and Ching, Y.T., The power of geometric duality revisited, *Inf. Process. Lett.* **21** (1985) 117–122.
40. Lee, D.T. and Preparata, F.P., Euclidean shortest paths in the presence of rectilinear boundaries, *Networks* **14** (1984) 393–410.
41. Lozano-Pérez, T. and Wesley, M.A., An algorithm for planning collision-free paths among polyhedral obstacles, *Comm. ACM* **22** (10) (1979) 560–570.
42. Lumelsky, V.J. and Stepanov, A.A., Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape, *Algorithmica* **2** (1987) 403–430.
43. Lyusternik, L.A., *Shortest Paths: Variational Problems* (Macmillan, New York, 1964).
44. Mehlhorn, K., *Multi-dimensional Searching and Computational Geometry*, EATCS Monographs on Theoretical Computer Science (Springer, New York, 1984).
45. Meng, A., Free-space modeling and geometric motion planning under unexpected obstacles, Tech. Rept., Texas Instruments Artificial Intelligence Lab (1987).
46. Meng, A., Free-space modeling and path planning under uncertainty for autonomous air robots, Tech. Rept., Texas Instruments Artificial Intelligence Lab (1987).
47. Metea, M.B. and Tsai, J.J.-P., Route planning for intelligent autonomous land vehicles using hierarchical terrain representation, in: *Proceedings IEEE International Conference on Robotics and Automation*, Raleigh, NC (1987).
48. Mitchell, J.S.B., An autonomous vehicle navigation algorithm, *Proc. SPIE Appl. Artif. Intell.* **485** (1984) 153–158.
49. Mitchell, J.S.B., Planning shortest paths, Ph.D. Thesis, Department of Operations Research, Stanford University, Stanford, CA (1986); also: Research Rept. 561, Artificial Intelligence Series, No. 1, Hughes Research Laboratories, Malibu, CA (1986).
50. Mitchell, J.S.B., Shortest rectilinear paths among obstacles, Tech. Rept. No. 739, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY (1987).
51. Mitchell, J.S.B., Shortest paths among obstacles, zero-cost regions, and roads, Tech. Rept. 764, Department of Operations Research, Cornell University, Ithaca, NY (1987).
52. Mitchell, J.S.B., On maximum flows in polyhedral domains, Extended Abstract, in: *Proceedings Fourth Annual ACM Conference on Computational Geometry*, Urbana-Champaign, IL (1988) 341–351.
53. Mitchell, J.S.B., On the maximum concealment problem, Tech. Rept., Department of Operations Research, Cornell University, Ithaca, NY (1988).
54. Mitchell, J.S.B., Mount, D.M. and Papadimitriou, C.H., The discrete geodesic problem, *SIAM J. Comput.* **16** (4) (1987) 647–668.
55. Mitchell, J.S.B. and Papadimitriou, C.H., Planning shortest paths, in: *Proceedings SIAM Conference on Geometric Modeling and Robotics*, Albany, NY (1985).
56. Mitchell, J.S.B. and Papadimitriou, C.H., The weighted region problem, Tech. Rept., Department of Operations Research, Stanford University, Stanford, CA (1986); Extended Abstract, in: *Proceedings Third Annual ACM Conference on Computational Geometry*, Waterloo, Ont. (1987); also: *J. ACM* (to appear).

57. Mitchell, J.S.B., Payton, D.W. and Keirsey, D.M., Planning and reasoning for autonomous vehicle control, *Int. J. Intell. Syst.* **2** (1987) 129–198.
58. Moravec, H.P., Obstacle avoidance and navigation in the real world by a seeing robot rover, Tech. Rept. CMU-RI-TR-3, Carnegie-Mellon Robotics Institute, Pittsburgh, PA (1980).
59. Nilsson, N.J., *Principles of Artificial Intelligence* (Tioga, Palo Alto, CA, 1980).
60. Oommen, B.J., Iyenger, S.S., Rao, S.V. and Kashyap, R.L., Robot navigation in unknown terrains using learned visibility graphs, Part I: The disjoint convex obstacle case, Tech. Rept. SCS-TR-86, School of Computer Science, Carleton University, Ottawa, Ont. (1986).
61. O'Rourke, J., Finding a shortest ladder path: A special case, Tech. Rept., Institute for Mathematics and its Applications, Preprint Series No. 353, University of Minnesota, Minneapolis, MN (1987).
62. O'Rourke, J., Suri, S. and Booth, H., Shortest paths on polyhedral surfaces, Manuscript, Johns Hopkins University, Baltimore, MD (1984).
63. Papadimitriou, C.H., An algorithm for shortest-path motion in three dimensions, *Inf. Process. Lett* **20** (1985) 259–263.
64. Papadimitriou, C.H., and Silverberg, E.B., Optimal piecewise linear motion of an object among obstacles, *Algorithmica* **2** (1987) 523–539.
65. Preparata, F.P. and Shamos, M.I., *Computational Geometry* (Springer, New York, 1985).
66. Quek, F.K.H., Franklin, R.F. and Pont, F., A decision system for autonomous robot navigation over rough terrain, in: *Proceedings SPIE Applications of Artificial Intelligence*, Boston, MA (1985)
67. Rao, N.S.V., Iyengar, S.S., Jorgensen, C.C. and Weisbin, C.R., Robot navigation in an unexplored terrain, *J. Rob. Syst.* **3** (4) (1986) 389–407.
68. Rao, N.S.V., Iyengar, S.S., Jorgensen, C.C. and Weisbin, C.R., On terrain acquisition by a finite-sized mobile robot in plane, in: *Proceedings IEEE International Conference on Robotics and Automation*, Raleigh, NC (1987) 1314–1319.
69. Reif, J.H. and Storer, J.A., Shortest paths in Euclidean space with polyhedral obstacles, Tech. Rept. CS-85-121, Computer Science Department, Brandeis University, Waltham, MA (1985).
70. Richbourg, R.F., Solving a class of spatial reasoning problems: Minimal-cost path planning in the Cartesian plane, Ph.D. Thesis, Naval Postgraduate School, Monterey, CA (1987).
71. Richbourg, R.F., Rowe, N.C. and Zyda, M.J., Exploiting capability constraints to solve global, two-dimensional path planning problems, Tech. Rept. NPS-86-006, Department of Computer Science, Naval Postgraduate School, Monterey, CA (1986).
72. Richbourg, R.F., Rowe, N.C., Zyda, M.J. and McGhee, R., Solving global two-dimensional routing problems using Snell's law and A* search, in: *Proceedings IEEE International Conference on Robotics and Automation*, Raleigh, NC (1987) 1631–1636.
73. Rowe, N.C., Roads, rivers, and rocks: Optimal two-dimensional route planning around linear features for a mobile robot, Tech. Rept. NPS52-87-027, Computer Science Department, Naval Postgraduate School, Monterey, CA (1987).
74. Rowe, N.C. and Richbourg, R.F., A new method for optimal path planning through nonhomogeneous free space, Tech. Rept. NPS52-87-003, Computer Science Department, Naval Postgraduate School, Monterey, CA (1987).
75. Rueb, K.D. and Wong, A.K.C., Structuring free space as a hypergraph for roving robot path planning and navigation, *IEEE Trans. Pattern Anal. Mach. Intell.* 9 (2) (1987) 263–273.
76. Samet, H. and Webber, R.E., Storing a collection of polygons using quadtrees, *ACM Trans. Graph.* **4** (3) (1985) 182–222.
77. Schwartz, J.T., Sharir, M. and Hopcroft, J. (Eds.), *Planning, Geometry, and Complexity of Robot Motion*, Ablex Series in Artificial Intelligence (Ablex, Norwood, NJ, 1987).
78. Schwartz, J.T. and Yap, C., (Ed.), *Algorithmic and Geometric Aspects of Robotics* **1** (Erlbaum, Hillsdale, NJ, 1987).
79. Sharir, M. and Schorr, A., On shortest paths in polyhedral spaces, *SIAM J. Comput.* **15** (1) (1986) 193–215.

80. Smith, T., Private communication, Department of Computer Science, University of California, Santa Barbara, CA (1986).
81. Souvaine, D., Computational geometry in a curved world, Ph.D. Thesis, Tech. Rept. CS-TR-094-87, Department of Computer Science, Princeton University, Princeton, NJ (1986).
82. Stern, H.I., A routing algorithm for a mobile search robot, in: S.I. Gass, H.J. Greenburg, K.L. Hoffman, and R.W. Langley (Eds.), *Impacts of Microcomputers on Operations Research* (North-Holland, Amsterdam, 1986) 73–87.
83. Sutherland, I.E., A method for solving arbitrary wall mazes by computer, *IEEE Trans. Comput.* **18** (12) (1969) 1092–1097.
84. Tarjan, R.E. and van Wyk, J., An $O(n \log \log n)$ algorithm for triangulating simple polygons, Tech. Rept., Bell Labs, Murray Hill, NJ (1986); also: *SIAM J. Comput.* (to appear).
85. Thorpe, C.E., Path relaxation: Path planning for a mobile robot, in: *Proceedings AAAI-84*, Austin, TX (1984) 318–321.
86. Thorpe, C.E., FIDO: Vision and navigation for a robot rover, Ph.D. Thesis, Tech. Rept. CMU-CS-84-168, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA (1984).
87. Welzl, E., Constructing the visibility graph for $n$ line segments in $O(n^2)$ time, *Inf. Process. Lett.* **20** (1985) 167–171.
88. Widmayer, P., Wu, Y.F. and Wong, C.K., Distance problems in computational geometry with fixed orientations, in: *Proceedings First Annual ACM Conference on Computational Geometry*, Baltimore, MD (1985).
89. Zikan, K., Private communication, Department of Operations Research, Stanford University, Stanford, CA (1986).

*Received October 1986; revised version received January 1988*