Annals of Mathematics and Artificial Intelligence, 3 (1991) 83–106

A NEW ALGORITHM FOR SHORTEST PATHS AMONG OBSTACLES IN THE PLANE

Joseph S.B. MITCHELL *

School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY 14853, USA

Abstract

We introduce a new algorithm for computing Euclidean shortest paths in the plane in the presence of polygonal obstacles. In particular, for a given start point s, we build a planar subdivision (a *shortest path map*) that supports efficient queries for shortest paths from s to any destination point t. The worst-case time complexity of our algorithm is $O(kn \log^2 n)$, where n is the number of vertices describing the polygonal obstacles, and k is a parameter we call the "illumination depth" of the obstacle space. Our algorithm uses $O(n)$ space, avoiding the possibly quadratic space complexity of methods that rely on visibility graphs. The quantity k is frequently significantly smaller than n, especially in some of the cases in which the visibility graph has quadratic size. In particular, k is bounded above by the number of different obstacles that touch any shortest path from s.

1. Introduction

Shortest path problems have been of interest in computational geometry for several years, due in a large part to their various applications in motion planning, visibility problems, terrain navigation, and wire routing. See [1,16,19,27] for several pointers to some relevant literature. A fundamental problem is that of finding shortest paths between two points in a space that is cluttered with "obstacles". Euclidean shortest paths among a collection of polygonal obstacles in the plane can be found in worst-case quadratic time (e.g., [26]), while the problem of finding shortest paths among polyhedral obstacles in three dimensions is known to be NP-hard [4].

In this paper, we present a new algorithm for computing Euclidean shortest paths in the plane among a set of disjoint polygonal obstacles bounded by n segments. Our algorithm actually computes the *shortest path map* (SPM) corresponding to a given start point s. An SPM is a planar subdivision which allows one to find the length of a shortest path to a query point in time $O(\log n)$ by point location, and to produce a shortest path in time $O(\log n + K)$, where K is the number of bends in the shortest path. See section 4 for more details.

* Partially supported by NSF Grants IRI-8710858 and ECSE-8857642 and by a grant from Hughes Research Laboratories, Malibu, CA.

The time complexity of our algorithm is $O(kn \log n + kV(n))$, where k is an output-sensitive quantity we call the *illumination depth* of the obstacle space, and $V(n)$ is the time to compute a geodesic Voronoi diagram of $O(n)$ points on the boundary of a simple polygon of size n. Currently, the best bound for $V(n)$ is $O(n \log^2 n)$, given by [2], yielding a running time of $O(kn \log^2 n)$ for our algorithm. The space complexity of our algorithm is $O(n)$, which compares favorably with many existing algorithms requiring up to quadratic space.

The illumination depth k is bounded above by the maximum number of different obstacles with which a shortest path from s comes in contact, which in turn is bounded above by the maximum number of bends in a shortest path from s. Frequently, though, k is much less than either of these estimates. In the worst case, $k = \theta(n)$, our algorithm performs slightly worse than the known $O(n^2)$ algorithms [3,9,13,22,23,26]; however, when k is small, our algorithm performs very well. If k is bounded, then our algorithm is nearly optimal.

If we are interested only in finding a shortest path from s to a fixed target point t, then our algorithm can be stopped as soon as the SPM it constructs first encounters t; in this case, k is bounded above by the *illumination depth of the point* t, which may be significantly less than the illumination depth of the entire obstacle space.

Our algorithm is unlike most previous shortest path algorithms in that it does not build a visibility graph; rather, we introduce a methodology which combines ideas from efficient visibility computation with ideas from shortest path map construction. We use a *partial* shortest path map, which does not include the effect of all obstacles, to trim down the amount of necessary visibility computation, and we use visibility considerations to simplify the computation of shortest path maps. In this way, we make a first step towards understanding the interplay between visibility graphs and shortest path maps.

2. Overview of the algorithm

We give a brief and informal summary of our algorithm. Precise definitions will be given later.

First, we compute the visibility polygon from the start point s. Then we build the shortest path map with respect to the set of "seen" obstacles that are wholly or partially visible from s. This is done in time $O(n \log^2 n)$ by reducing the problem to that of computing a Voronoi diagram within an appropriately defined simple polygon and then appealing to the algorithm of [2].

Next, by applying the results of [11] for finding a face in an overlay of two arrangements, we compute the region of "accessibility" corresponding to the subset of "seen" obstacles that define our current shortest path map. We say that a point p is accessible if there exists a shortest path from s to p that does not bend at a vertex of an "unseen" obstacle. Obstacles that are on the boundary of

the region of accessibility and that were previously "unseen" are now considered to be "seen".

We now extend the shortest path map to include the effect of the newly "seen" obstacles by defining appropriate geodesic Voronoi diagram problems on simple polygons, and appealing again to the results of [2]. We continue iterating this process, computing accessibility and then extending the shortest path map, until all obstacles have been "seen". The number of iterations is called the *illumination depth*, k, of the obstacle space.

3. A brief survey of existing algorithms

Many algorithms have been developed to find shortest paths among obstacles in the plane. We survey in table 1 the algorithms known to us at this time, together with their worst-case space and time bounds. A less current but more detailed survey is provided in [16]. All algorithms except (7) and (11) proceed by building a visibility graph (or a subgraph thereof) and then searching it using Dijkstra's algorithm [7] or A* [21].

The table includes only those algorithms that apply to general sets of disjoint polygonal obstacles in the plane. Various special cases have faster algorithms. For example, the case in which all obstacles are convex was considered by [16] and [24]. The case of vertical line segment obstacles was solved in optimal time by [15], and this result was generalized to the case of obstacles with disjoint projections onto some given line [16]. An important special case is that of finding shortest paths within a simple polygon (without holes), which can be solved in linear time for a triangulated simple polygon [10].

We have restricted our attention to the case of Euclidean shortest paths. The problem of finding shortest paths according to the L_1 metric (or any fixed

Table 1
A comparison of shortest path algorithms

Alg.	Space $S(n)$	Time $T(n)$	Ref.
(1)	E_{VG}	n^3	[28]
(2)	E_{VG}	$n^2 \log n$	[14,25]
(3)	E_{VG}	$n^2 \log m$	
(4)	n	$n^2 \log m$	
(5)	n	$Kn \log m$	
(6)	n^2	n^2	[3,26]
(7)	n	$nm + n \log n$	[23]
(8)	E_{VG}	$E_{VG} + n \log n$	[9]
(9)	n	$E_{VG} \log n$	[22]
(10)	E_{SP}	$E_{SP} + n \log n$	[13]
(11)	n	$kn \log^2 n$	

orientation metric) has been solved in nearly optimal time by [6] and [17]. These results, and others, have also led to efficient approximation algorithms for finding paths whose length is within a constant factor of the true geodesic distance [5,17].

Our notation is as follows: the obstacle space consists of m disjoint polygonal obstacles bounded by a total of n segments, $E_{VG} \leqslant \binom{n}{2}$ is the number of edges in the visibility graph, and E_{SP} is the number of edges in the visibility graph that are locally tangent at both endpoints, meaning that these edges are sufficient for searching for shortest paths [13,16]. (An edge $\overline{uv}$ is *locally tangent* at v if the line through $\overline{uv}$ is tangent to the set $B_\epsilon(v) \cap obst(v)$ for some $\epsilon > 0$, where $B_\epsilon(v)$ is the ball of radius ϵ about v and $obst(v)$ is the obstacle at vertex v.) The quantities K and k are defined below.

The main result we present in this paper is algorithm (11) (see table 1). A few words are in order about the other bounds that are listed in table 1 without references to the literature:

(3) The time bound of $O(n^2 \log m)$ is a result of the simple observation that the visibility polygon with respect to a fixed point of a set of m simple polygonal obstacles can be computed in time $O(n \log m)$ [3].

(4) We can reduce the space complexity of algorithm (3) by doing the visibility calculations "on the fly" while running Dijkstra's algorithm. The idea is simply not to build the entire visibility graph and then search it, but rather to compute the visibility from a vertex only at the moment when the vertex is about to be "expanded" by Dijkstra's algorithm. We compute the visibility from the vertex, update the labels on the vertices found to be adjacent to it, and discard those edges that did not result in label improvement. The result of this is that we need to keep a data structure of only linear size.

(5) This complexity bound is a result of the trivial observation that it suffices to terminate the search for a shortest path to a prespecified target point t once t has been permanently labeled. Here, K is the number of nodes expanded by Dijkstra's algorithm before t is reached.

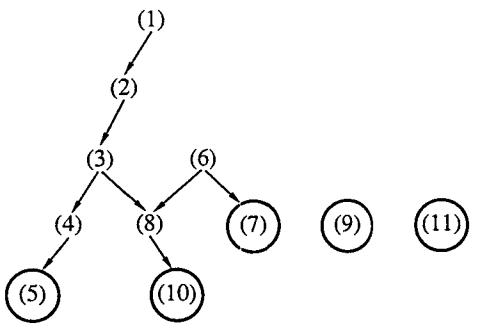

Fig. 1. Dominance graph of shortest path algorithms.

(11) This is the main algorithm described in this paper. Here, k is the "illumination depth" of the obstacle space, as defined in section 8. Frequently, k will be bounded by a small number, in which case this algorithm performs very well.

It is difficult to say which of the many techniques will be best in practice. We can, however, give the relationships among the worst-case running times and space requirements. We will say that algorithm (i) is *dominated* by algorithm (j) if (j) has both space and time bounds that are as good as or better than those of algorithm (i). We can then construct the dominance graph, as shown in fig. 1. Algorithms (5), (7), (9), (10), and (11) (see table 1) are undominated, meaning that there are cases in which each of them is best, either in terms of space or in terms of time complexity.

4. Notation and preliminaries

The input to our algorithm will be $(\mathcal{F}, s)$, where $\mathcal{F}$ is a closed multiply-connected polygonal domain in the plane, and $s \in \mathcal{F}$ is the *start* (or *source*) point. We refer to $\mathcal{F}$ as *free space*, and we let V denote the set of n vertices of $\mathcal{F}$. For simplicity of presentation, we will assume that $\mathcal{F}$ is bounded; our results extend easily to the unbounded case. Thus, $\mathcal{F}$ consists of a simple polygon P, minus a set of m disjoint polygonal holes (*obstacles*).

We let $\mathcal{O}$ denote the *obstacle space*, which is the complement of the free space $\mathcal{F}$. Thus, $\mathcal{O}$ consists of m (open, bounded) simple polygonal holes, and the (unbounded) complement of the simple polygon P.

For any two points $p, q \in \mathcal{F}$, let $g(p, q)$ denote a geodesic (i.e., shortest obstacle-avoiding) path from p to q, and let $d(p, q)$ be its Euclidean length. When we need to emphasize the set $\mathcal{O}$ of obstacles with respect to which the path is geodesic, we will write $g_{\mathcal{O}}(p, q)$ and $d_{\mathcal{O}}(p, q)$, and we call the path *geodesic*$(\mathcal{O})$. Note that, while $d(p, q)$ is always well-defined and unique, there may, in general, be many geodesic paths from p to q.

It is well-known (e.g., [14]) that geodesic paths in polygonal domains are polygonal paths with turn points at vertices of the domain. We say that $r \in V$ is a *root* of $p \in \mathcal{F}$ if, for some geodesic path $g(s, p)$, r is the last vertex along $g(s, p) \setminus \{p\}$ at which $g(s, p)$ turns. If the line segment $\overline{sp}$ is a geodesic path, then s is a root of p. The set of all roots of p is denoted by $\mathcal{R}(p)$.

In the remainder of this paper, we make the following *general position assumption*: for any $\mathcal{O}' \subseteq \mathcal{O}$, and for any vertex $v \in V$, there is a unique geodesic$(\mathcal{O}')$ path from s to v. With this assumption, every vertex $v \in V$ has a unique root, which we denote by *root*(v). Our algorithm can be modified to handle the general case, but the details are tedious.

The *shortest path tree*, SPT$(s, \mathcal{O})$, *with respect to point* s and obstacle space $\mathcal{O}$ is the tree whose nodes are the vertices $v \in V$ and whose edges link each node v to *root*(v). With our general position assumption, SPT$(s, \mathcal{O})$ is unique.

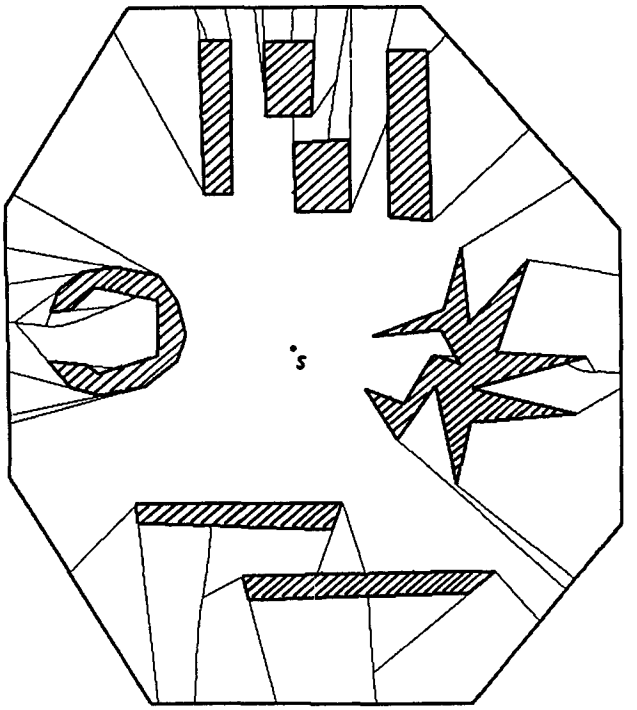

Fig. 2. A shortest path map.

While SPT(s, $\mathcal{O}$) gives the tree of shortest paths from s to every vertex, a "shortest path map" gives a description of the set of shortest paths from s to all points of free space. The *shortest path map*, SPM(s, $\mathcal{O}$), *with respect to point s* and obstacle space $\mathcal{O}$ is a partition of $\mathcal{F}$ 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, SPM(s, $\mathcal{O}$) is the partitioning of $\mathcal{F}$ into cells $C(\mathcal{R}) = \mathrm{P}\{x \in \mathcal{F} \mid \mathcal{R} = \mathcal{R}(x)\}$ corresponding to subsets $\mathcal{R} \subseteq V \cup \{s\}$. If $\mathcal{R} = \{v\}$ is a singleton, we write $C(v)$ to denote the cell of the SPM *rooted at* vertex v.

In general, $C(\mathcal{R})$ will not be connected. If $\mathcal{R} = \{v\}$ is a singleton, then it is easy to show that $C(v)$ is two-dimensional and connected. In particular, the cell $C(s)$ rooted at s is simply the visibility polygon about s with respect to $\mathcal{O}$. If $\mathcal{R} = \{v_i, v_j\}$ is a pair, then one can show that $C(\mathcal{R})$ is one-dimensional and possibly disconnected, and we call $C(\{v_i, v_j\})$ the *bisector* of vertices v_i and v_j. If $\mathcal{R}$ has cardinality at least three, then $C(\mathcal{R})$ is either empty or a single point, called an *SPM-vertex*. We denote the set of all bisectors and SPM-vertices by $\mathcal{B}$. An example of an SPM is shown in fig. 2.

We define a vertex v to be *locally tangent* if the line segment joining *root*(v) and v is locally tangent at v, as defined in section 3. Note that the segment joining *root*(v) and v is necessarily locally tangent at *root*(v). It is easy to see

that if v is not locally tangent, then the cell $C(v)$ is empty, while if v is locally tangent, then the cell $C(v)$ will be nonempty.

For each vertex v, we can define the *extension segment rooted at* v as follows: extend a ray from the point $root(v)$ through v until the point y where it first leaves $\mathscr{F}$. The segment $\overline{vy}$ is the *extension segment rooted at* v. If v is not locally tangent, then the extension segment is just the point v.

We state now a few simple facts about bisectors and shortest path maps.

FACT 4.1

Each bisector is the union of a finite set of closed subarcs of a common hyperbola. (A straight line is considered to be a degenerate case of a hyperbola.)

Proof

This follows from basic analytic geometry. □

FACT 4.2

Each cell $C(v)$ of an SPM is star-shaped with respect to v. Furthermore, the boundary of $C(v)$ consists of straight line segments, which arise as subsegments of obstacle boundaries or as subsegments of extension segments rooted at v, and hyperbolic arcs, which arise as the bisectors between v and the roots of cells neighboring $C(v)$.

Proof

This follows directly from fact 4.1 and the definition of an SPM. □

FACT 4.3

No geodesic path from s can cross a bisector or go through an SPM-vertex.

Proof

Assume that $g(s, t)$ does cross a bisector at a point x. (The case of going through an SPM-vertex is handled similarly.) There must be another path $g'(s, x)$ that is shortest but which has a different root than that of the subpath of $g(s, t)$ from s to x. Now path $g(s, t)$ cannot bend at point x (otherwise, it could be shortened). Also, the path we get by appending $g'(s, x)$ to the subpath of $g(s, t)$ from x to t must be a shortest path from s to t, so it too cannot bend at point x. This is inconsistent with the fact that the two paths have different roots. □

FACT 4.4

$\mathscr{B}$ forms a forest.

Proof

If not, then there must be a cycle. Take any point t separated from s by the cycle and note that any geodesic path to it must cross a bisector or go through an SPM-vertex, violating fact 4.3. □

FACT 4.5

There is at least one bisector point on the boundary of each obstacle.

Proof

Let $x \in \partial O$ be a point of obstacle O that is closest to s (in geodesic distance) and fix a geodesic path $g(s, x)$. Now parameterize the points on the boundary ∂O according to the arc length from x to each point, going clockwise about ∂O. The parameter (call it θ) ranges from 0 to S, where S is the total length of the perimeter of O.

We say that a path Π from s to $y(\theta)$ *goes around O on the right* if the region enclosed by the following cycle contains O: go from s to $y(\theta)$ along Π, then go from $y(\theta)$ to x counterclockwise along the perimeter of O, then go back to s along (the reverse of) $g(s, x)$. We similarly define the notion of going around O on the left.

A shortest path from s to any $y(\theta)$ will go around O either on the left or on the right. Let $F(\theta)$ (resp., $G(\theta)$) be the length of the shortest path from s to point $y(\theta) \in \partial O$ that goes around O on the right (resp., left). For θ sufficiently close to 0, the shortest path to $y(\theta)$ will go around O on the right, while for θ sufficiently close to S, it will go around O on the left. Functions F and G are continuous, so by the intermediate value theorem, there must be a crossing point $y(\theta^*)$ on the boundary of O. Such a crossing point must correspond to a bisector point. □

FACT 4.6

$\mathcal{O} \cup \mathcal{B}$ is simply connected.

Proof

Suppose that $\mathcal{O} \cup \mathcal{B}$ is not connected. Then, there must be a maximal connected component C of the set $\mathcal{O} \cup \mathcal{B}$ that is not connected to the boundary of P, the polygon that forms the outer boundary of free space $\mathcal{F}$. No geodesic path crosses C, since C is composed of obstacles and bisectors (refer to fact 4.3). Thus, we may think of C as if it were an "obstacle". Following the proof of fact 4.5, with trivial modifications to allow for the curved boundary of C, we know that there is another bisector incident on the boundary of C that separates geodesics that go around C on the right from those that go around C on the left. This contradicts the maximality of the component C.

The fact that $\mathcal{O} \cup \mathcal{B}$ is simply connected follows from the same reasoning as in fact 4.4: if $\mathcal{O} \cup \mathcal{B}$ had a hole, then points x within the hole are in free space, but any path from s to x must either cross an obstacle or a bisector. Thus, by fact 4.3, there can be no geodesic path from s to a point x in a hole, contradicting the fact that there must be a geodesic path from s to every point of free space, since free space is assumed to be connected. □

Remark

Note that if we had not assumed $\mathcal{F}$ to be bounded, then fact 4.6 remains true if we connect all unbounded bisector arcs to a common point at infinity.

FACT 4.7

$\mathcal{F} \setminus \mathcal{B}$ is a simply connected region.

Proof

This follows from fact 4.6 by noting that if the set were multiply connected, then its holes would be connected components of $\mathcal{O} \cup \mathcal{B}$. Another way to see this is by noting that any path within $\mathcal{F} \setminus \mathcal{B}$ joining two points of $\mathcal{F} \setminus \mathcal{B}$ can be contracted to the point s by the continuous mapping that carries points along their geodesic paths to s. □

FACT 4.8

An SPM is a planar map of combinatorial size $O(n)$.

Proof

Consider the planar dual of an SPM in which a node is associated with each two-dimensional cell. If the boundaries of cells $C(v)$ and $C(v')$ intersect, then draw an edge between node v and node v' for each connected component of the intersection $\partial C(v) \cap \partial C(v')$. Edges of this graph are of two varieties: those that are dual to extension segments, and those that are dual to arcs of bisectors. There are only $O(n)$ extension segments, since there is only one per vertex. The only remaining issue is that there may be many edges between the same two nodes v and v', since a bisector $C(\{v, v'\})$ may have many connected components. But between any two edges joining nodes v and v' there must be an obstacle, so we can charge off these multiple occurrences to the obstacles, yielding an overall linear bound on the size of the graph. □

We use data structures such as that of [12] to store shortest path maps (and other planar maps), so that basic operations, such as traversing a cell's boundary, can be done efficiently. Furthermore, applying the methods of [8], a subdivision of size N can be processed in time $O(N \log N)$ to support $O(\log N)$ point location queries. Then, from fact 4.8, we can conclude

FACT 4.9

Given SPM(s, $\mathcal{O}$), one can, with $O(n \log n)$ preprocessing time, construct a data structure of size $O(n)$ such that for any query point t, one can determine the length of a shortest path from s to t in time $O(\log n)$ and can produce a shortest path $g(s, t)$ in additional time $O(B)$, where $B = O(n)$ is the number of bends in the path $g(s, t)$.

Proof

By locating point t in the subdivision SPM, we can identify in $O(\log n)$ time a root r on a shortest path $g(s, t)$. Since the SPM can store distances $d(s, v)$ for all vertices v, and we know the root vertex r for t, we simply compute $d(s, t) = d(s, r) + |rt|$. We can output a path by following the sequence of root pointers from r back to s. □

5. Defining accessibility

The usual definition of visibility with respect to an "obstacle" set $\mathcal{O} \subset \mathbb{R}^d$ is that points p and q are *visible* if and only if the line segment joining them does not intersect $\mathcal{O}$. A more general notion of visibility can be defined in terms of any pair of distance functions, as discussed in an earlier draft of this paper [18]. For our purposes here, we use a notion of generalized visibility that we call "accessibility".

Given two sets of obstacles, $\mathcal{O}_1$ and $\mathcal{O}_2$, we say that two points $p, q \in \mathbb{R}^2 \setminus \{\mathcal{O}_1 \cup \mathcal{O}_2\}$ are *accessible with respect to* $\mathcal{O}_1$ *modulo* $\mathcal{O}_2$ if and only if there exists a geodesic($\mathcal{O}_1$) path from p to q that does not intersect $\mathcal{O}_2$. The usual notion of visibility among a set of obstacles $\mathcal{O}$ is the special case in which $\mathcal{O}_1 = \varnothing$, $\mathcal{O}_2 = \mathcal{O}$. We use the mnemonic of the subscript "1" on the *primary* set of obstacles $\mathcal{O}_1$ and the subscript "2" on the *secondary* set of obstacles $\mathcal{O}_2$. The primary obstacles are those that define the geodesic metric, while the secondary obstacles are those that define what is hidden.

We let $\mathcal{V}_s(\mathcal{O}_1, \mathcal{O}_2)$ denote the set of all points accessible from s with respect to $\mathcal{O}_1$ modulo $\mathcal{O}_2$. Thus $\mathcal{V}_s(\mathcal{O}_1, \mathcal{O}_2)$ is the union of the points on all geodesic($\mathcal{O}_1$) paths from s that do not intersect $\mathcal{O}_2$. Note that the set $\mathcal{V}_s(\varnothing, \mathcal{O})$ is just the (usual) visibility polygon of s among obstacles $\mathcal{O}$. Figure 3 shows an example of the accessibility region $\mathcal{V}_s(\mathcal{O}_1, \mathcal{O}_2)$: the primary obstacles $\mathcal{O}_1$ are shown with diagonal hatching, the secondary obstacles $\mathcal{O}_2$ are shown with cross hatching, and the inaccessibility region is shown shaded with dots. Note that points that lie in cells of the SPM whose roots are not accessible from s are necessarily not accessible from s.

We define a set $A \subseteq \mathbb{R}^2 \setminus \mathcal{O}$ to be *$\mathcal{O}$-star-shaped about* $s \in A$ if for every $x \in A$ there exists a geodesic($\mathcal{O}$) path π from s to x with $\pi \subseteq A$. Finally, we define an obstacle O to be *accessible* from s (with respect to $\mathcal{O}_1$ modulo $\mathcal{O}_2$) if there exists a point $x \in O$ such that x is accessible from s (with respect to $\mathcal{O}_1$ modulo $\mathcal{O}_2$).

FACT 5.1

$\mathcal{V}_s(\mathcal{O}_1, \mathcal{O}_2)$ is precisely the maximal star-shaped (with respect to the geodesic($\mathcal{O}_1$) metric) subset of $\mathbb{R}^2 \setminus \{\mathcal{O}_1 \cup \mathcal{O}_2\}$.

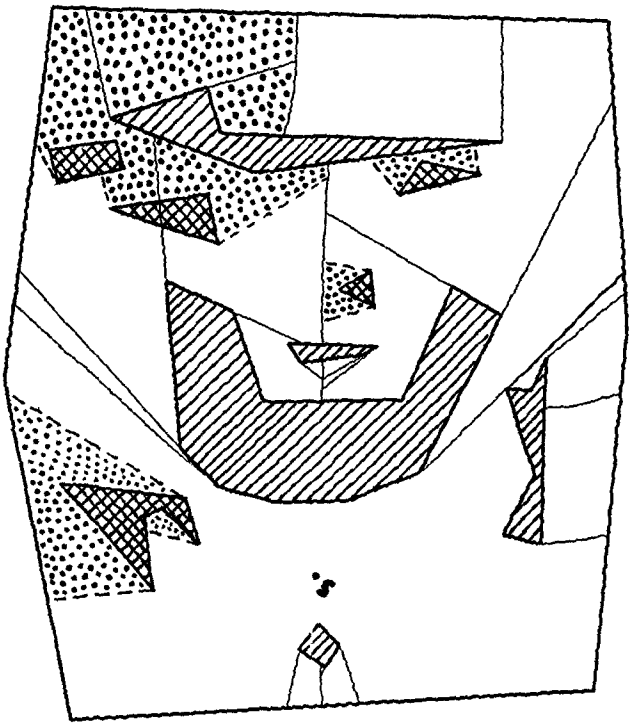

Fig. 3. Accessibility region $\mathcal{V}_s(\mathcal{O}_1, \mathcal{O}_2)$.

Proof

This follows directly from the definition of accessibility. □

FACT 5.2

The boundary of $\mathcal{V}_s(\mathcal{O}_1, \mathcal{O}_2)$ consists of $O(n_1 + n_2)$ straight line segments (which arise as subsegments of obstacle boundaries or as subsegments of extension segments rooted at vertices of $\mathcal{O}_2$) and hyperbolic arcs (which arise as subarcs of bisectors present in SPM(s, $\mathcal{O}_1$)). Here, n_1 and n_2 are the number of vertices describing $\mathcal{O}_1$ and $\mathcal{O}_2$, respectively.

Proof

This fact will be shown in the proof of theorem 6.1 in the following section, when we prove correctness of the algorithm that computes the region $\mathcal{V}_s(\mathcal{O}_1, \mathcal{O}_2)$. □

6. Computing accessibility regions

We consider in this section the problem of computing the accessibility region $\mathcal{V}_s(\mathcal{O}_1, \mathcal{O}_2)$. We will assume that we are given SPM(s, $\mathcal{O}_1$), the shortest path map

with respect to s and obstacle space $\mathcal{O}_1$. Let n_i be the number of vertices in obstacle space $\mathcal{O}_i$ $(i = 1, 2)$.

In an early draft [18], we gave a means of computing $\mathcal{V}_s(\mathcal{O}_1, \mathcal{O}_2)$ in time $O((n_1 + n_2)\log(n_1 + n_2))$ by sweeping a geodesic($\mathcal{O}_1$) path about s, in a manner very similar to the standard method of computing a visibility profile by an angular sweep (see [14]). The method relied on judiciously "breaking" the obstacles $\mathcal{O}_2$ at selected points on the boundary, so as to assure that the "lower envelope" can be determined. We omit the discussion of this method here, since we will present instead a very simple means of obtaining the same time bound using some recent results on computing arrangements of curves [11]. We are grateful to one of the referees for devising the method presented below.

ALGORITHM A

Input: Point s, obstacle spaces $\mathcal{O}_i$ for $i = 1, 2$ (vertex set V_i of size n_i), and the shortest path map SPM(s, $\mathcal{O}_1$).

Output: A description of the boundary of $\mathcal{V}_s(\mathcal{O}_1, \mathcal{O}_2)$ in terms of a sequence of straight line segments and hyperbolic arcs.

(0) Locate each of the vertices $v \in V_2$ of the secondary obstacles in the map SPM(s, $\mathcal{O}_1$). This can be done in time $O(n_2 \log n_1)$, assuming that we have already done the $O(n_1 \log n_1)$ preprocessing on SPM(s, $\mathcal{O}_1$) so that it can support point location queries (see [8]). Let r_v be the root for vertex v. For each v, produce a *pseudo-extension segment* extending from v away from r_v until it hits the boundary of $C(r_v)$. This can be done in $O(n_2 \log n_2 + n_1 + n_2)$ time by sorting the $\mathcal{O}_2$ vertices about their respective roots, and then scanning about the roots to determine where the pseudo-extension segments terminate on the cell boundaries.

(1) Let $\mathcal{M}_1$ be the arrangement formed by the primary obstacle boundaries, the bisector set $\mathcal{B}$ corresponding to SPM(s, $\mathcal{O}_1$), and the pseudo-extension segments produced in step (0). This arrangement has complexity $O(n_1 + n_2)$ (since the SPM has complexity $O(n_1)$ and the pseudo-extension segments were constructed in such a way as not to cross bisectors of primary obstacle boundaries). In particular, the face of $\mathcal{M}_1$ that contains s will have linear complexity, and we can assume we have a boundary description of the face.

(2) Let $\mathcal{M}_2$ be the arrangement formed by the boundaries of the secondary obstacles $\mathcal{O}_2$. This arrangement has size $\mathcal{O}(n_2)$, and, again, we can assume we have a boundary description of the face containing s.

(3) Consider the overlay arrangement, $\mathcal{A}$, one gets by superimposing $\mathcal{M}_1$ and $\mathcal{M}_2$. We prove below (theorem 6.1) that the face of $\mathcal{A}$ that contains s is precisely the set $\mathcal{V}_s(\mathcal{O}_1, \mathcal{O}_2)$. The Combination Lemma of [11] implies that the face containing s in the overlay arrangement $\mathcal{A}$ has complexity $O(n_1 + n_2)$. The

merge algorithm of [11] finds the face containing s in time $O(\log(n_1 + n_2))$ times the size of the input plus output (which is $O(n_1 + n_2)$ in our case), yielding an overall bound of $O((n_1 + n_2)\log(n_1 + n_2))$ to produce a boundary description of $\mathcal{V}_s(\mathcal{O}_1, \mathcal{O}_2)$.

THEOREM 6.1

Algorithm A computes the accessibility region $\mathcal{V}_s(\mathcal{O}_1, \mathcal{O}_2)$ in time $O((n_1 + n_2)\log(n_1 + n_2))$.

Proof

The time complexity follows immediately from the statement of the algorithm above. Thus, we will be done with the proof of the theorem, as well as a proof of fact 5.2, once we show that the cell containing s in $\mathcal{A}$ is indeed the region $\mathcal{V}_s(\mathcal{O}_1, \mathcal{O}_2)$.

Consider a point p in free space. If $p \in \mathcal{V}_s(\mathcal{O}_1, \mathcal{O}_2)$, then there exists a geodesic$(\mathcal{O}_1 \cup \mathcal{O}_2)$ path from s to p that does not bend at a vertex of $\mathcal{O}_2$. This implies that there exists a root $r \in \mathcal{R}(p)$ for p in SPM$(s, \mathcal{O}_1)$ such that $\overline{rp}$ does not intersect $\mathcal{O}_2$ and the unique geodesic path $g_{\mathcal{O}_1}(s, r)$ from s to r does not intersect $\mathcal{O}_2$. Since the path from s to r (along $g_{\mathcal{O}_1}(s, r)$) to p (along $\overline{rp}$) does not cross any obstacle boundaries of $\mathcal{O}_1$ and $\mathcal{O}_2$ and does not cross any pseudo-extension segments or bisectors, the point p must lie in the cell of $\mathcal{A}$ containing s.

Conversely, if p lies in the same cell with s in $\mathcal{A}$, then the segment $\overline{rp}$ does not intersect $\mathcal{O}_2$ for some root $r \in \mathcal{R}(p)$, since otherwise p would be disconnected from s by the union of pseudo-extension segments and obstacle boundaries of $\mathcal{O}_2$. Similarly, the segment joining r to its (unique) root $root(r)$ must not cross $\mathcal{O}_2$, since r must be in the same cell as s and p. Thus, $g_{\mathcal{O}_1}(s, r)$ must lie in the cell containing s, showing that p is indeed accessible from s. □

7. Extending the shortest path map

A simple consequence of the definition of accessibility is the fact that the accessibility region is correctly subdivided by the current shortest path map:

LEMMA 7.1

The partitioning of $\mathcal{V}_s(\mathcal{O}_1, \mathcal{O}_2)$ according to SPM$(s, \mathcal{O}_1)$ is exactly the same as the partitioning of $\mathcal{V}_s(\mathcal{O}_1, \mathcal{O}_2)$ according to SPM$(s, \mathcal{O}_1 \cup \mathcal{O}_2)$.

Proof

Consider any $t \in \mathcal{V}_s(\mathcal{O}_1, \mathcal{O}_2)$. If t lies in cell $C(v)$ of SPM$(s, \mathcal{O}_1)$, then it is visible to v (fact 4.2). *Before* the addition of the secondary obstacles $\mathcal{O}_2$, the shortest path from s to t went from s to v along a geodesic$(\mathcal{O}_1)$ path, and then went from v to t along $\overline{vt}$. The fact that t is accessible implies that this path is

still feasible after the addition of the secondary obstacles $\mathcal{O}_2$. Clearly, adding the secondary obstacles $\mathcal{O}_2$ cannot make this path any shorter, so t must lie in cell $C(v)$ of SPM(s, $\mathcal{O}_1 \cup \mathcal{O}_2$).

Conversely, if t is an accessible point that lies in cell $C(v)$ of SPM(s, $\mathcal{O}_1 \cup \mathcal{O}_2$), then the geodesic($\mathcal{O}_1 \cup \mathcal{O}_2$) path from s to t with root v must have no vertices of $\mathcal{O}_2$ along it, so it is also a geodesic($\mathcal{O}_1$) path. Thus, t lies in cell $C(v)$ in the map SPM(s, $\mathcal{O}_1$). □

The problem we consider in this section is that of extending the shortest path map into the region which is *not* accessible.

We begin by identifying those secondary obstacles that are fully hidden from view; they may be ignored for the moment, since they will be incorporated into the shortest path map in a later iteration of the main algorithm. We concentrate only on the set $\mathcal{O}_a \subseteq \mathcal{O}_1 \cup \mathcal{O}_2$ of obstacles that are accessible with respect to $\mathcal{O}_1$ modulo $\mathcal{O}_2$. Given the boundary description of $\mathcal{V}_s(\mathcal{O}_1, \mathcal{O}_2)$, it is easy to identify the set $\mathcal{O}_a$ by traversing the boundary of $\mathcal{V}_s(\mathcal{O}_1, \mathcal{O}_2)$, noting those obstacles that have vertices or edges in common with the boundary.

We let $\mathcal{U} = P \setminus \{\mathcal{V}_s(\mathcal{O}_1, \mathcal{O}_2) \cup \mathcal{O}_a\}$ be the set of inaccessible ("unseen") points, including those points that lie in obstacles that do not border the accessible region. From fact 5.2 we know that the boundary of $\mathcal{U}$ consists of straight line segments (which are subsegments of obstacle edges or extension segments) and hyperbolic arcs (which are subarcs of the bisectors in the original shortest path map). If no point on the boundary of P is accessible, then $\mathcal{U}$ is a connected region R_0 with a single hole. Otherwise, $\mathcal{U}$ is a union of a set of simply connected regions, $R_1, \ldots, R_J$.

The basic idea of our algorithm is to transform the problem of building the shortest path map partition of $\mathcal{U}$ into that of building a geodesic Voronoi diagram of an appropriate set of sources in an appropriate set of simple polygons.

The geodesic Voronoi diagram of a set of point sites in a simple polygon P is defined in precisely the same manner as the usual Voronoi diagram of a set of point sites in the plane, except that the distance function used is that of geodesic distance within the polygon. Thus, the interior of the polygon is partitioned into cells according to the lengths of Euclidean shortest paths to the sites.

Geodesic Voronoi diagrams for a set of point sites in a simple polygon are discussed in [2], where an $O(n \log^2 n)$ algorithm for constructing such diagrams is given. In fact, Aronov [2] builds the "shortest path partition" of the geodesic Voronoi diagram, in which each Voronoi cell is further subdivided according to the last vertex along the unique shortest path from the cell's site to the points of the cell.

Before giving details of our algorithm, we make a few important remarks concerning our application of the results of [2]:

(1) Aronov [2] requires a general position assumption in order for the algorithm to work as he states. Our general position assumption is strong enough to

imply his. It is possible, but tedious, to remove the assumption from the algorithm of [2].

(2) In the case that the unseen region $\mathcal{U}$ consists of a single connected component with one hole, we will need to compute a geodesic Voronoi diagram within a polygon with a single hole. We can still apply the algorithm of [2], with a minor modification. First, we make a cut along a chord joining the hole to the outer boundary of the polygon. In the resulting simple polygon, we compute the diagram by applying [2]. We then must use the extend and merge procedures of [2] to obtain the correct diagram in the original polygon with the hole.

(3) The sites in our Voronoi problems will always occur on the boundary of the enclosing polygon. We think of the sites as being "just inside" the polygon.

(4) We will need to compute *weighted* geodesic Voronoi diagrams in which there is an additive nonnegative weight associated with each site. More precisely, this means that the distance from a point p to a site r is defined to be the geodesic distance from p to r *plus* the weight of r. It is easy to check that the resulting weighted geodesic Voronoi diagram has cells with straight or hyperbolic boundaries. It is also not difficult to modify the algorithm of [2] to handle the case of weights without affecting the worst-case running time. Instead of presenting the details here, we can refer the reader to an alternate approach that we presented in an earlier draft of this paper [18], where we showed that the weighted geodesic Voronoi problems that arise in our application can be solved directly by the method of [2] as follows. First we note that the sites in our application will always be on the boundary of the polygon. Then, for each site r, we construct a very skinny straight corridor (a "leg") extending from r with length equal to the weight of r. We then consider there to be a source (without weight) at the end of the leg. Although the legs may intersect the polygon, this is not important in being able to run the algorithm of [2] on the new polygon; we can think of the legs as lying in different Riemann sheets so that they do not interfere with the execution of the algorithm.

ALGORITHM B

Input: Point s, the shortest path map SPM(s, $\mathcal{O}_1$), a description of the boundary of $\mathcal{V}_s(\mathcal{O}_1, \mathcal{O}_2)$.

Output: A subdivision of the set of unseen points, $\mathcal{U} = P \setminus \{\mathcal{V}_s(\mathcal{O}_1, \mathcal{O}_2) \cup \mathcal{O}_a\}$, according to the shortest path map SPM(s, $\mathcal{O}_a$), where $\mathcal{O}_a \subseteq \mathcal{O}_1 \cup \mathcal{O}_2$ is the set of obstacles that are accessible with respect to $\mathcal{O}_1$ modulo $\mathcal{O}_2$.

(0) By traversing the boundary of SPM(s, $\mathcal{O}_1$), determine the set $\mathcal{O}_a \subseteq \mathcal{O}_1 \cup \mathcal{O}_2$ of obstacles that are accessible with respect to $\mathcal{O}_1$ modulo $\mathcal{O}_2$.

(1) Construct the planar map describing the boundary of the set of unseen points, $\mathcal{U} = P \setminus \{\mathcal{V}_s(\mathcal{O}_1, \mathcal{O}_2) \cup \mathcal{O}_a\}$. $\mathcal{U}$ is either a single connected region R_0 with a single hole ($\mathcal{V}_s(\mathcal{O}_1, \mathcal{O}_2) \cup \mathcal{O}_a$) or a union of simply connected regions, $R_1, \ldots, R_J$.

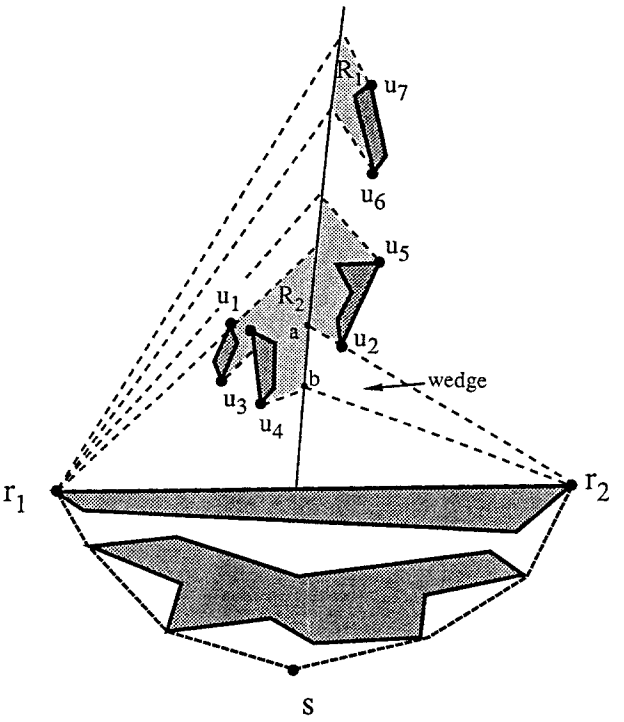

Fig. 4. Adding wedges to R_1 and R_2.

(2) Traverse the boundary of $\mathcal{U}$, keeping the set $\mathcal{U}$ on the right, replacing each hyperbolic arc ab with the pair of line segments $\overline{ar}$ and $\overline{rb}$, where r is the root whose cell $C(r)$ lies to the left of arc ab. The result of adding "wedges" in this way is a new set, $\mathcal{U}' \supseteq \mathcal{U}$, with straight boundary segments. We refer to the roots r that form the apices of the wedges as *wedge roots*. In fig. 4, the hyperbolic arc ab is replaced by line segments $\overline{ar_2}$ and $\overline{r_2 b}$, which bound a wedge with wedge root r_2. Note that there may be many wedge roots occupying the same location; we assume that separate instances are created for each. In fig. 4, the original regions R_1 and R_2 are shown shaded, and there are three wedges (with dashed boundaries), with two wedge roots at location r_1 and one wedge root at location r_2. By lemma 7.2 below, the set $\mathcal{U}'$ will either be a polygon Q_0 with a single polygonal hole (in the case that $\mathcal{U}$ was a single component R_0 with a hole) or a set of disjoint simple polygons, $Q_1, \ldots, Q_J$ (in the case that $\mathcal{U}$ is a union of simple regions $R_1, \ldots, R_J$).

(3) Identify the set of *shadow-producing vertices*: a vertex u of $\mathcal{O}_2$ is shadow-producing if it is accessible (with respect to $\mathcal{O}_1$ modulo $\mathcal{O}_2$) from s, and u is not on the boundary of a wedge. In fig. 4, points u_i, $i = 3, \ldots, 7$, are shadow-producing vertices (two other vertices, u_1 and u_2, are not shadow-producing since they lie on wedge boundaries).

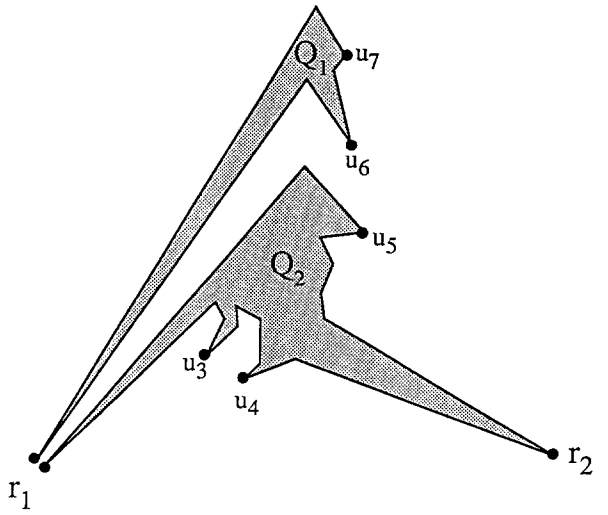

Fig. 5. The polygonal regions Q_1 and Q_2.

(4) For each connected component Q_j of $\mathcal{U}'$, define an instance of a weighted geodesic Voronoi problem (WVD_j) within Q_j in which the set S_j of sites for WVD_j is the set of all shadow-producing vertices and wedge roots on the boundary of Q_j, with each site weighted by its geodesic distance from s (which is known from SPM(s, $\mathcal{O}_1$)). Figure 5 illustrates the two Voronoi problems corresponding to the polygons Q_1 and Q_2 that arise from fig. 4; the sites are $S_1 = \{r_1, u_6, u_7\}$ and $S_2 = \{r_1, r_2, u_3, u_4, u_5\}$.

LEMMA 7.2

The process of enlarging $\mathcal{U}$ by adding wedges in step (2) yields a new region $\mathcal{U}'$ that is either a polygon Q_0 with a single polygonal hole (in the case that $\mathcal{U}$ has one connected component R_0 with a single hole) or a set of disjoint simple polygons $Q_1, \ldots, Q_J$ (in the case that $\mathcal{U}$ is the union of simple regions $R_1, \ldots, R_J$).

Proof

Consider the wedge w bounded by $\overline{rb}$, arc ba, and $\overline{ar}$. Since w is a subset of the cell $C(r)$, it is free of primary ($\mathcal{O}_1$) obstacles. It must also be free of secondary ($\mathcal{O}_2$) obstacles, since otherwise arc ab could not have been on the boundary of a connected component of $\mathcal{U}$. Since the shortest path map is a planar subdivision, no other wedge could intersect the interior of w. (There may be other wedges with wedge root at position r.) Thus, the wedges we add to $\mathcal{U}$ to obtain $\mathcal{U}'$ are nonoverlapping, implying that the connected components of $\mathcal{U}'$ are also nonoverlapping. Note too that we have not created any holes by the addition of nonoverlapping wedges. Furthermore, $\mathcal{U}'$ has only straight boundary segments. □

LEMMA 7.3

Solving the weighted Voronoi diagram problem WVD_j yields a subdivision of R_j identical to that of SPM(s, $\mathcal{O}_a$) restricted to R_j.

Proof

A geodesic path from s to point $t \in R_j$ must enter Q_j either through a wedge root or through a shadow-producing vertex, since a geodesic path cannot cross a bisector (fact 4.3) or relative interior of the portion $\overline{rx}$ of an extension segment that bounds an SPM cell (since points on $\overline{rx}$ can be reached by a shortest path with root r, by definition of the cell $C(r)$). Thus, the SPM partitioning of Q_j is equivalent to the weighted Voronoi diagram partitioning (with the refinement of the shortest path partitioning) of P_j in which the weights assigned to source points equal their distances from s with respect to obstacle space $\mathcal{O}_a$. By our construction, WVD_j solves precisely this problem. □

THEOREM 7.4

Algorithm B correctly extends the shortest path map into the set $\mathcal{U} = P \setminus \{\mathcal{V}_s(\mathcal{O}_1, \mathcal{O}_2) \cup \mathcal{O}_a\}$ of unseen points and its running time is $O(V(n_1 + n_2))$, where $V(n) = O(n \log^2 n)$ is the time required to solve a weighted geodesic Voronoi diagram problem of size n within a simple polygon.

Proof

Correctness was established in the previous lemma. Since $\mathcal{V}_s(\mathcal{O}_1, \mathcal{O}_2)$ and SPM(s, $\mathcal{O}_1$) are of linear size, the total input size of all of the Voronoi diagram problems WVD_j is $O(n_1 - n_2)$. □

8. The main algorithm

ALGORITHM C

Input: Point s, a set $\mathcal{O}$ of m disjoint polygonal obstacles within an enclosing simple polygon P, with a total of n edges bounding P and the obstacles.

Output: The shortest path map SPM(s, $\mathcal{O}$).

(0) Let $\mathcal{O}_1 = \varnothing$, and let $\mathcal{O}_2 = \mathcal{O}$. Let $k = 0$. Initially, let SPM be the trivial planar subdivision consisting of a single cell corresponding to the entire polygon P.
(1) If $\mathcal{O}_2 = \varnothing$, then stop. Otherwise, go on to step (2).
(2) Compute $\mathcal{V}_s(\mathcal{O}_1, \mathcal{O}_2)$ using algorithm A of section 6. This requires $O(n \log n)$ time. Let $\mathcal{O}_a$ be the set of all obstacles that are accessible from s with respect to $\mathcal{O}_1$ modulo $\mathcal{O}_2$, i.e., those that have part of their boundary in common with the boundary of $\mathcal{V}_s(\mathcal{O}_1, \mathcal{O}_2)$.

(3) Extend the SPM subdivision into the set of unseen points $\mathcal{U}$ using algorithm B of section 7. This requires $O(V(n)) = O(n \log^2 n)$ time and yields SPM(s, $\mathcal{O}_a$).
(4) Set $\mathcal{O}_1 \leftarrow \mathcal{O}_a$, $\mathcal{O}_2 \leftarrow \mathcal{O}_2 \setminus \mathcal{O}_a$, and $k \leftarrow k + 1$. Go to step (1).

Remarks

(a) Note that we only place accessible obstacles in the set $\mathcal{O}_1$, so there will never be a primary $\mathcal{O}_1$ obstacle that is not accessible with respect to $\mathcal{O}_1$ modulo $\mathcal{O}_2$; thus, it will always be the case that $\mathcal{O}_1 \subset \mathcal{O}_a$.

(b) The first time that we enter step (2) of the algorithm, we compute $\mathcal{V}_s(\varnothing, \mathcal{O})$, which is simply the visibility polygon about s with respect to all obstacles. While this can be done using algorithm A, one would usually prefer to do this in time $O(n \log n)$ by a standard visibility sweep about s.

(c) If we are interested only in finding a shortest path from s to a destination point t, given in advance, then step (1) of the above algorithm can be modified to stop as soon as t becomes accessible. This may greatly reduce the number of iterations necessary.

(d) The following simple observation can potentially lead to a considerable improvement in the running time when a destination point t is given in advance. For a given point t, we notice that it suffices to extend the shortest path map only into the component, R_j, of $\mathcal{U}$ that contains t, not into the entire set $\mathcal{U}$. This saves us the effort of solving many Voronoi diagram problems each time we call algorithm B. A related observation was made in [16], where it was shown that one only needs to consider the component of the "cone of overlapping obstacle profiles" that contains a given destination t.

The value of k at the conclusion of the algorithm records the total number of major iterations required until termination. If a destination point t is given in advance, then k is called the *illumination depth* of t and is denoted by $k(t)$;

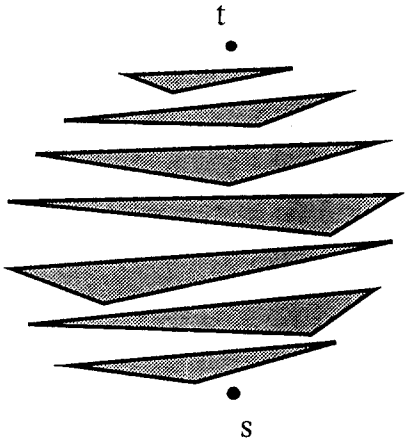

Fig. 6. A case in which $k(\mathcal{O}) = \Omega(n)$.

otherwise, $k = \max_{t \in \mathcal{F}} k(t)$ is called the *illumination depth* of the obstacle space and is denoted by $k(\mathcal{O})$. In the worst case, $k(\mathcal{O})$ can be $\Omega(n)$ (see fig. 6), but it will frequently be much smaller than n. For example, if we know that s can see at least part of the boundary of every obstacle, then $k(\mathcal{O}) = 1$. We can show a simple upper bound on $k(\mathcal{O})$.

FACT 8.1

$k(t)$ is bounded above by the number of different obstacles in contact with any geodesic path from s to t. In particular, $k(t)$ (and, hence, $k(\mathcal{O})$) is bounded above by m, the number of obstacles.

Proof

Let $g(s, t)$ be any geodesic path from s to t. Let $O_1, \ldots, O_M$ be the set of obstacles touched by $g(s, t)$, in order of increasing distance from s to x_j, the first point of O_j encountered by $g(s, t)$. With each major iteration of our algorithm, at least one new obstacle becomes accessible. The algorithm terminates once O_M becomes accessible. This implies that $k(t) \leqslant M$. □

COROLLARY 8.2

$k(t)$ is bounded above by the number of edges in any geodesic path from s to t.

In conclusion, we have the following:

THEOREM 8.3

The shortest path map of a polygonal obstacle space of size n can be found in time $O(kn \log n + kV(n))$, where $k = k(\mathcal{O})$ is the ilumination depth of the space and $V(n)$ is the time required to compute the geodesic Voronoi diagram of a set of $O(n)$ sites within a simple polygon of size n. The current bound of $V(n) = O(n \log^2 n)$ yields an overall bound of $O(kn \log^2 n)$.

9. Dynamic shortest path map maintenance

Our algorithm can also be used in a natural way to maintain a shortest path map, given a sequence of insertions and deletions of obstacles. In the dynamic problem, one would like to avoid doing a complete recomputation of the shortest path map if the change induced by the insertion/deletion is minor.

First, note that if we want to insert one or more obstacles into an SPM, we may end up altering the diagram in a very significant way. For example, placing just one obstacle in such a way as to cross an edge of the current shortest path tree results in modifying the shortest path map involving all of the vertices in the subtree that was cut off. So the size of the change can be $\Omega(n)$.

We can accommodate an insertion by applying our algorithm as follows: Assume that we have SPM(s, $\mathcal{O}$), and we are given a new set of obstacles, $\mathcal{O}'$, to add to our obstacle space. First, we apply algorithm A to compute $\mathcal{V}_s(\mathcal{O}, \mathcal{O}')$. Then, by a traversal of the boundary of $\mathcal{V}_s(\mathcal{O}, \mathcal{O}')$, we determine the set $\mathcal{O}_a \subseteq \mathcal{O} \cup \mathcal{O}'$ of accessible obstacles with respect to $\mathcal{O}$ modulo $\mathcal{O}'$. Now, we simply apply algorithm C, starting at step (3) with $\mathcal{O}_1 = \mathcal{O}_a$ and $\mathcal{O}_2 = (\mathcal{O} \cup \mathcal{O}') \setminus \mathcal{O}_a$. The running time is bounded by $O(kn \log^2 n)$, where k is now the number of major iterations that we must do in order to correct for the addition of the new obstacles. One may expect that k will be small in many cases, but it can be $\Omega(n)$ in the worst case.

Just as an insertion can cause a linear-size change in the SPM, so can a deletion. In order to handle a deletion of a set $\mathcal{O}' \subseteq \mathcal{O}$ of obstacles, we proceed as follows. We are given SPM(s, $\mathcal{O}$). The points that are not accessible with respect to $\mathcal{O} \setminus \mathcal{O}'$ modulo $\mathcal{O}'$ are those points x that have roots $\mathcal{R}(x)$ that are vertices of $\mathcal{O}'$ or descendents of vertices of $\mathcal{O}'$ in the shortest path tree SPT(s, $\mathcal{O}$), since such points x have a vertex of $\mathcal{O}'$ on every shortest path from s to x. Thus, we first traverse the boundaries of all cells rooted at vertices of the deleted obstacles or at descendants of these vertices in SPT(s, $\mathcal{O}$). This allows us to construct the accessible region $\mathcal{V}_s(\mathcal{O} \setminus \mathcal{O}', \mathcal{O}')$. Let $\mathcal{O}_1$ be the accessible obstacles (those that appear on the boundary of $\mathcal{V}_s(\mathcal{O} \setminus \mathcal{O}', \mathcal{O}')$), and let $\mathcal{O}_2 = \mathcal{O} \setminus \mathcal{O}_1$. We now apply algorithm C starting at step (3). Again, the running time is bounded by $O(kn \log^2 n)$, where k is now the number of major iterations required to correct for the deletion of $\mathcal{O}'$.

10. Conclusion

There are a variety of algorithms known for computing shortest paths, but at the present time there is no one algorithm that is fastest in every case. We have presented here a new technique which is an improvement over previous algorithms in many cases and is nearly optimal ($O(n \log^2 n)$) in cases in which the illumination depth k is bounded. Unfortunately, there are situations in which our algorithm is worse than existing $O(n^2)$ solutions to the shortest path problem. Several open questions remain:

(1) Can the geodesic Voronoi diagram of a set of source points in a simple polygon be computed in time $O(n \log n)$? If the current bound of $V(n) = O(n \log^2 n)$ were to be improved, it would imply a corresponding improvement in the time bound of our algorithm. Also, it may be possible to take advantage of the special structure present in our case. In particular, how efficiently can one find the geodesic Voronoi diagram of a set of sources all of which are on the polygon boundary?

(2) Can the running time of our algorithm be improved to something like $O(kn + V(n))$? Such an improvement would reduce the worst-case running time

of our algorithm to quadratic. It is not likely that such an improvement will be a trivial extension to our algorithm, since we require $O(n \log n)$ time at each iteration just to do the point location queries in algorithm A.

(3) Can we get an algorithm with running time $O(Ln \log^2 n)$, where L is the minimum number of line segments in any obstacle-free polygonal path from s to t? (L is called the "link distance" from s to t.) We know that the illumination depth k is bounded above by the number of links in any shortest path from s to t, but it is not true in general that $k \leqslant L$.

(4) Can we combine the ideas presented here with those of other existing algorithms to obtain an algorithm that is optimal or nearly optimal in more cases? In particular, some of the cases that are bad for our algorithm are quite easy for existing algorithms (e.g., the visibility graph may be sparse, as in fig. 6). Loosely speaking, it seems that our algorithm works well in many cases in which the visibility graph is dense and poorly in some cases in which it is sparse.

Acknowledgements

The author would like to thank Erik Wynters for his implementation work which provided the complex shortest path map pictures, Esther Arkin for useful discussions on this research, and the participants of the research seminar on computational geometry at Cornell for many stimulating conversations. The referees provided many useful suggestions for improvements of the paper. One referee is responsible for many corrections to the original paper and suggesting the current version of algorithm A, using the methods of [11]. This research was partially supported by National Science Foundation grants ECSE-8857642 and IRI-8710858, and by a grant from the Hughes Artificial Intelligence Center, Hughes Research Laboratories, Malibu, CA.

References

[1] H. Alt and E. Welzl, Visibility graphs and obstacle-avoiding shortest paths, Z. Oper. Res. 32 (1989) 145–164.

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

[3] T. Asano, T. Asano, L.J. Guibas, J. Hershberger and H. Imai, Visibility of disjoint polygons, Algorithmica 1 (1986) 49–63.

[4] J. Canny and J. Reif, New lower bound techniques for robot motion planning problems, *Proc. 28th FOCS* (Oct. 1987) pp. 49–60.

[5] K. Clarkson, Approximation algorithms for shortest path motion planning, *Proc. 19th Annual ACM Symp. on Theory of Computing*, New York City (May 25–27, 1987) pp. 56–65.

[6] K. Clarkson, S. Kapoor and P. Vaidya, Rectilinear shortest paths through polygonal obstacles in $O(n \log^2 n)$ time, *Proc. 3rd Annual ACM Conf. on Computational Geometry*, Waterloo, Ontario (1987) pp. 251–257 (to appear in Algorithmica).

[7] Dijkstra, A note on two problems in connection with graphs, Num. Mathematik 1 (1959) 269–271.
[8] H. Edelsbrunner, L.J. Guibas and J. Stolfi, Optimal point location in a monotone subdivision, SIAM J. Comput. 15 (1986) 317–340.
[9] S.K. Ghosh and D.M. Mount, An output sensitive algorithm for computing visibility graphs, *Proc. 28th Annual IEEE Symp. on Foundations of Computer Science* (1987) pp. 11–19.
[10] L.J. Guibas, J. Hershberger, D. Leven, M. Sharir and R. Tarjan, Linear time algorithms for visibility and shortest path problems inside triangulated simple polygons, Algorithmica 2 (1987) 209–233.
[11] L.J. Guibas, M. Sharir and S. Sifrony, On the general motion planning problem with two degrees of freedom, Discr. Comput. Geom. 4 (1989) 491–521.
[12] L.J. Guibas and J. Stolfi, Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams, ACM Trans. Graphics 4 (1985) 74–123.
[13] S. Kapoor and S.N. Maheshwari, Efficient algorithms for Euclidean shortest path and visibility problems with polygonal obstacles, *Proc. 4th Annual ACM Symp. on Computational Geometry*, Urbana-Champaign, IL (June 6–8, 1988) pp. 172–182.
[14] D.T. Lee, Proximity and reachability in the plane, Ph.D. Thesis, Technical Report ACT-12, Coordinated Science Laboratory, University of Illinois (Nov. 1978).
[15] D.T. Lee and F.P. Preparata, Euclidean shortest paths in the presence of rectilinear boundaries, Networks 14 (1984) 393–410.
[16] J.S.B. Mitchell, Planning shortest paths, PhD Thesis, Department of Operations Research, Stanford University (August, 1986).
[17] J.S.B. Mitchell, Shortest rectilinear paths among obstacles, Technical Report No. 739, School of Operations Research and Industrial Engineering, Cornell University (April, 1987) (revised version to appear in Algorithmica).
[18] 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).
[19] J.S.B. Mitchell, An algorithmic approach to some problems in terrain navigation, Art. Intell. 37 (1988) 171–201.
[20] J.S.B. Mitchell, D.M. Mount and C.H. Papadimitriou, The discrete geodesic problem, SIAM J. Comput. 16 (4) (1987) 647–668.
[21] N.J. Nilson, *Principles of Artificial Intelligence* (Tioga Publ. Co., Palo Alto, CA, 1980).
[22] M.H. Overmars and E. Welzl, New methods for computing visibility graphs, *Proc. 4th Annual ACM Symp. on Computational Geometry*, Urbana-Champaign, IL (June 6–8, 1988) pp. 164–171.
[23] 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).
[24] H. Rohnert, Time and space efficient algorithms for shortest paths between convex polygons, IPL 27 (1988) 175–179.
[25] M. Sharir and A. Schorr, On shortest paths in polyhedral spaces, SIAM J. Comput. 15 (1) (1986) 193–215.
[26] E. Welzl, Constructing the visibility graph for n line segments in $O(n^2)$ time, Inf. Processing Lett. 20 (1985) 167–171.
[27] C.K. Yap, Algorithmic motion planning, in: *Advances in Robotics: Vol. 1*, eds. J.T. Schwartz and C.K. Yap (Lawrence Erlbaum Assoc., 1987) pp. 95–143.
[28] G.E. Wangdahl, S.M. Pollock and J.B. Woodward, Minimum-trajectory pipe routing, J. Ship Res. 18 (1974) 46–49.