SIAM J. COMPUT.
Vol. 16, No. 4, August 1987

THE DISCRETE GEODESIC PROBLEM*

JOSEPH S. B. MITCHELL†, DAVID M. MOUNT‡ AND CHRISTOS H. PAPADIMITRIOU§

Abstract. We present an algorithm for determining the shortest path between a source and a destination on an arbitrary (possibly nonconvex) polyhedral surface. The path is constrained to lie on the surface, and distances are measured according to the Euclidean metric. Our algorithm runs in time $O(n^2 \log n)$ and requires $O(n^2)$ space, where n is the number of edges of the surface. After we run our algorithm, the distance from the source to any other destination may be determined using standard techniques in time $O(\log n)$ by locating the destination in the subdivision created by the algorithm. The actual shortest path from the source to a destination can be reported in time $O(k+\log n)$, where k is the number of faces crossed by the path. The algorithm generalizes to the case of multiple source points to build the Voronoi diagram on the surface, where n is now the maximum of the number of vertices and the number of sources.

Key words. shortest paths, computational geometry, geodesics, Dijkstra's algorithm

AMS(MOS) subject classification. 68E99

1. Introduction. Recent research in the algorithmic aspects of robot motion and terrain navigation has resulted in a number of interesting variants of the shortest path problem. The problem of determining the shortest path between two points in three dimensions in the presence of polyhedral obstacles seems to be very hard, and only extremely inefficient algorithms [16], [17] or approximation algorithms [14] are known for it. The two-dimensional special case with polygonal obstacles can be solved easily in $O(n^2 \log n)$ time by constructing the *visibility graph* [4], [6], [8], [9], [17] and $O(n \log n)$ is attainable in special cases when shortest paths possess certain monotonicity properties [5], [9], [17] (here, n is the total number of vertices in the polygonal obstacles). Recently, an algorithm for the two-dimensional shortest path problem which runs in time $O(nk+n \log n)$ (where k is the number of disjoint simple polygonal obstacles) has been announced [16].

In this paper we examine a special case of the three-dimensional shortest path problem called the *Discrete Geodesic Problem.* We are given two points which lie on the surface of a given polyhedron, and we are asked to find the shortest path between them which lies on the polyhedral surface. The geodesic problem is of considerable interest in terrain navigation, where a moving vehicle is bound to move along a surface that could be modeled by a polyhedron. Notice that this measure of distance may not be the shortest distance in three-space which avoids the polyhedron, since we are constrained to travel along the surface. To see why this is a special case of the three-dimensional problem, consider two polyhedral obstacles, namely the open polyhedron and the complement of its closure; a path between two points which avoids both polyhedra is exactly a path on the polyhedral surface. On the other hand, the

* Received by the editors April 29, 1985; accepted for publication (in revised form) December 5, 1986. This research was partially supported by a grant from the Hughes Aircraft Company.

† Department of Operations Research, Stanford University and Hughes Artificial Intelligence Center, Stanford, California 94305. This author was supported by a Howard Hughes Doctoral Fellowship and is affiliated with the Hughes Artificial Intelligence Center, Calabasas, California.

‡ Department of Computer Science, University of Maryland, Baltimore, Maryland 21201. This author was partially supported by the Air Force Office of Scientific Research under contract F-49620-83-C-0082.

§ Department of Operations Research and Department of Computer Science, Stanford University, Stanford, California 94305.

problem we solve is a generalization of the two-dimensional problem, since any polygonal obstacle can be simulated by a tall prism emanating from the plane whose base is the given polygon.

The shortest path problem on a surface was first posed in [17], where an $O(n^3 \log n)$ algorithm was given for the case of *convex* polyhedra. Here n is a measure of the complexity of the scene, which in our case we may take as the number of edges, say, of the polyhedral surface. Mount [10] has given an algorithm for convex polyhedra which improves the running time to $O(n^2 \log n)$. For *nonconvex* polyhedra, [13] gives an $O(n^5)$ algorithm.

We improve these results as follows: We handle general (possibly nonconvex, and even of higher genus) polyhedra, and we improve the time bound to $O(n^2 \log n)$. Furthermore, we solve the single-source shortest path problem (and by a simple extension, the multiple-source shortest path problem; see [11]), creating a subdivision for a given source point so that we can find the length of the shortest path to any destination point simply by locating it in the subdivision. Thus, in $O(\log n)$ time, the length of the shortest path is determined to any other destination, and the shortest path can then be listed in time $O(k)$, where k is the number of edges crossed by the path. Our algorithm therefore generalizes the work in [4] and [17] for the two-dimensional shortest path problem among obstacles. We concentrate here on the case of *bounded* polyhedra; however, the generalization of the algorithm to handle unbounded faces is straightforward.

Our algorithm uses a technique we call "continuous Dijkstra," as it closely resembles in structure the famous algorithm of Dijkstra for finding shortest paths in a graph [1]. Edges of the polyhedron behave like nodes of a graph, except that here there is no unique distance from the source to an edge. Instead, there is a *function* that serves as a label for a node, and we keep track of a discrete description of the minimum such function. This involves keeping track of "intervals of optimality" on each edge, which subdivide the edge into regions for which the shortest path to points in the region have the same discrete structure, passing through the same sequence of vertices and edges. These intervals are similar to the notion of "slices" used in [17].

2. Preliminaries.

We are given a polyhedral surface, $\mathcal{S}$, specified by a set of faces, 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 two special points s and t (the *source* and *destination*, respectively). Without loss of generality, we assume that all faces are triangles (since polygons may be triangulated in time $O(n \log n)$ [2] and the number of edges introduced is linear in the number of vertices) and that s and t are vertices of the polyhedral surface. We are asked to find the shortest path from the source to the destination which lies completely on the surface.

DISCRETE GEODESIC PROBLEM (DGP).

Instance: Two points s and t on a polyhedral surface $\mathcal{S}$, which has been triangulated.

Question: Find a shortest path in the Euclidean metric from s to t such that the path stays on the surface $\mathcal{S}$.

In fact, our algorithm solves the following query form of the problem.

SINGLE-SOURCE DISCRETE GEODESIC PROBLEM (SS-DGP).

Instance: Two points s and t on a polyhedral surface $\mathcal{S}$, which has been triangulated.

Question: Build a structure which allows one to compute a shortest path from s to any query point t such that the path stays on the surface $\mathcal{S}$.

Thus, we are able to find the shortest path between the source and *any* point on the surface, and we know of no algorithm which solves DGP in less time (worst-case, asymptotic) than SS-DGP. Consequently, in our exposition we shall not concern ourselves with a specific destination point.

Two faces f and f' are said to be *edge-adjacent* if they share a common edge e. A *sequence of edge-adjacent faces* is a list of one or more faces $\mathcal{F} = (f_1, f_2, \cdots, f_{k+1})$ such that face f_i is edge-adjacent to face f_{i+1} (sharing common edge e_i) (see Fig. 1). We then refer to the (possibly empty) list of edges $\mathcal{E} = (e_1, e_2, \cdots, e_k)$ as an *edge sequence* and to the vertex of face f_1 which is opposite e_1 as the *root*, $r(\mathcal{E})$, of $\mathcal{E}$ ($r(\emptyset)$ is undefined). If no face (resp., edge) appears more than once in $\mathcal{F}$ (resp., $\mathcal{E}$), then we call the sequence *simple*.

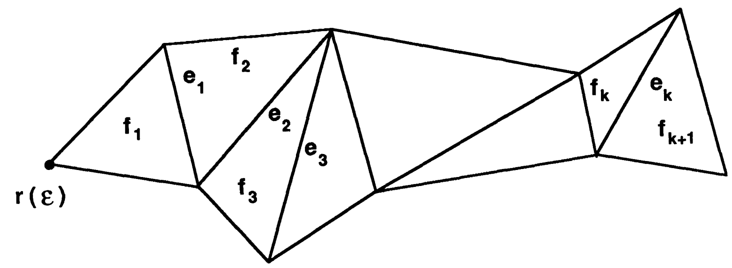

FIG. 1. *A sequence of edge-adjacent faces.*

Each face has a two-dimensional coordinate system associated with it. If faces f and f' are edge-adjacent sharing edge e, we define the *planar unfolding of face f' onto face f* as the image of points of f' when rotated about the line through e into the plane of f such that the points of f' fall on the *opposite* side of e to points of f (i.e., do not unfold face f' *on top of* face f). Points in the planar unfolding of f' onto f are written in the coordinate system of face f. Extending this notion, we say that we *unfold* an edge sequence $\mathcal{E} = (e_1, e_2, \cdots, e_k)$ as follows: Rotate f_1 around e_1 until its plane coincides with that of f_2, rotate f_1 and f_2 around e_2 until their plane coincides with that of f_3, continue in this way until all faces $f_1, f_2, \cdots, f_k$ lie in the plane of f_{k+1}. The resulting *planar unfolding along $\mathcal{E}$* represents points of faces $f_1, f_2, \cdots, f_k$ in the coordinate system of face f_{k+1}. If x is a point on face f_i $(1 \leqq i \leqq k+1)$, then let $U_{\mathcal{E}}(x)$ be the *unfolded image of x along $\mathcal{E}$* (written in the coordinate system of face f_{k+1}).

A *path* on $\mathcal{S}$ will always be understood to be a *simple* (that is, not self-intersecting) path whose intersection with any face is a union of disjoint line segments. A *geodesic path* is a path which is locally optimal and cannot, therefore, be shortened by slight perturbations. We say that path p *connects* edge sequence $\mathcal{E} = (e_1, e_2, \cdots, e_k)$ if p consists of segments which join interior points of $e_1, e_2, \cdots, e_k$ (in that order). (Note that such a path cannot pass through the endpoint of any e_i, for otherwise two distinct consecutive edges of $\mathcal{E}$ would have to be collinear.) Path p *goes through* edge sequence $\mathcal{E}$ if it has a subpath which connects $\mathcal{E}$ (note that it may have many such subpaths). If p connects edge sequence $\mathcal{E}$, then the *planar unfolding of p along $\mathcal{E}$* is simply the unfolded image of p along $\mathcal{E}$.

3. Characterization of geodesics and optimal paths.

We begin a characterization of geodesics and optimal paths with a simple existence lemma stating that geodesics and optimal paths *exist* (but they may not be *unique*).

LEMMA 3.1. *There exists a geodesic path from s to any other point $x \in \mathcal{S}$. Furthermore, among the geodesic paths from s to x there exists at least one of minimal length.*

We have assumed that paths are simple, so they do not go through any point more than once. (This is certainly true for shortest paths.) Optimal paths also have the property that they do not go through any face more than once, as the following lemma states.

LEMMA 3.2. *The intersection of an optimal path p with a face f is a (possibly empty) line segment.*

Proof. If $p \cap f \neq \emptyset$, then let y be the *first* point of p on f, and let y' be the *last.* Then the part of p from y to y' must also be optimal (otherwise, p could be improved). The shortest path on $\mathcal{S}$ from y to y' is the line segment connecting them. Thus, the intersection of p with f must be a line segment. □

Note that it is possible for the intersection of a *geodesic* path with a face to be a union of many disjoint line segments. For example, if one tightly wraps a string many times around a long thin rectangular bar, the resulting path of the taut string is locally optimal but clearly not globally optimal. This also shows that not all geodesics are optimal paths and that there can be infinitely many geodesic paths (although only finitely many of them can pass through each face at most once).

We now state a simple property of geodesics (observed by [7] and [17]) which says that if one unfolds a geodesic path along an edge sequence, it becomes a straight line segment.

LEMMA 3.3. *If p is a geodesic path which connects the edge sequence $\mathcal{E}$, then the planar unfolding of p along the edge sequence $\mathcal{E}$ is a straight line segment.*

Proof. If the image were *not* a straight line segment, then there exist segments $\overline{\alpha x}$ and $\overline{x\alpha'}$ of p such that x is interior to edge $e = f \cap f' \in \mathcal{E}$ and the angle $\angle \alpha x \alpha'$ is not equal to π. Then, there exists a "shortcut" $\overline{\beta\beta'}$ for p, with $\beta \in \alpha x$, $\beta' \in x\alpha'$, and segment $\overline{\beta\beta'}$ lies entirely within the faces f and f'. This contradicts the local optimality of p. □

In the case of convex polyhedra, an optimal path will pass through no vertices other than its endpoints, and this property plays a critical role in the algorithm of [17]. In the case of nonconvex polyhedra, however, an optimal path may pass through many other vertices. For example, in Fig. 2 the optimal path from s to t goes through vertex v. (Vertex v has the property that the sum of the angles at v on the faces adjacent to v is greater than 2π.) The optimal path may, in fact, follow a long chain of adjacent edges (see Fig. 3).

We can characterize the way in which a geodesic path passes through a vertex. If path p passes through vertex v while going from face f to face f', then let $\overline{\alpha v}$ be the segment of p on f leading into v, and let $\overline{v\alpha'}$ be the segment of p on f' leading

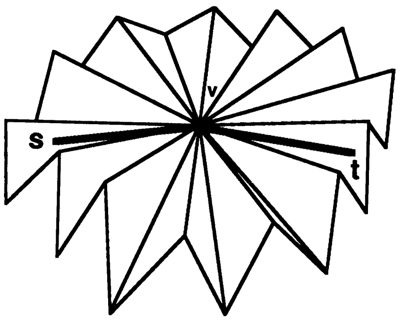

FIG. 2. *An optimal path going through a vertex.*

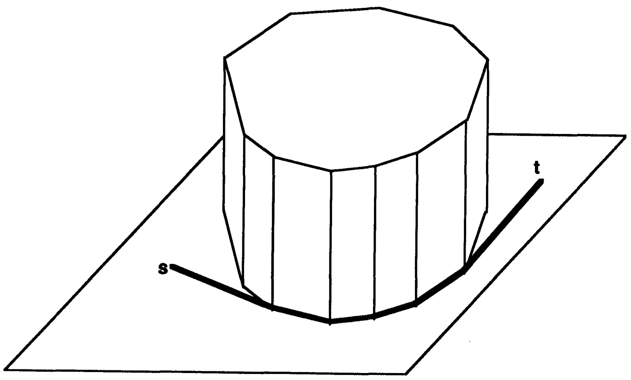

FIG. 3. *An optimal path following a chain of edges.*

out of v. Define the *clockwise* angle of p at v as the angle *from* $\overline{\alpha v}$ *to* $\overline{v\alpha'}$ when the faces containing v from f to f' (inclusive) *in a clockwise order about* v are "flattened" (simply add the angles formed by the edges incident on v; this may give you an angle far greater than 2π). Similarly, define the *counterclockwise* angle from $\overline{\alpha v}$ to $\overline{v\alpha'}$. Define the *angle made by path p through vertex v* as the minimum of these two angles. We then have the following.

LEMMA 3.4. *If a geodesic path p passes through vertex v, then the angle made by p at v is* $\geqq \pi$.

Proof. Assume that the angle is *less than* π and that (without loss of generality) it is clockwise. When the faces $f_1, f_2, \cdots, f_k$ are flattened about v, we get the situation depicted in Fig. 4. Now note that there exists a point β on the segment $\overline{\alpha v}$ and a point β' on the segment $\overline{v\alpha'}$ such that the segment $\overline{\beta\beta'}$ lies entirely in the set of flattened faces $f, f_1, f_2, \cdots, f_k, f'$. Clearly, the segment $\overline{\beta\beta'}$ represents a "short cut" for the path through v, so p cannot be a geodesic. □

The subpath of p between any two consecutive vertices v and v' connects some edge sequence $\mathscr{E}$ with root $v = r(\mathscr{E})$ (if $\mathscr{E} = \varnothing$, then v and v' are the endpoints of some

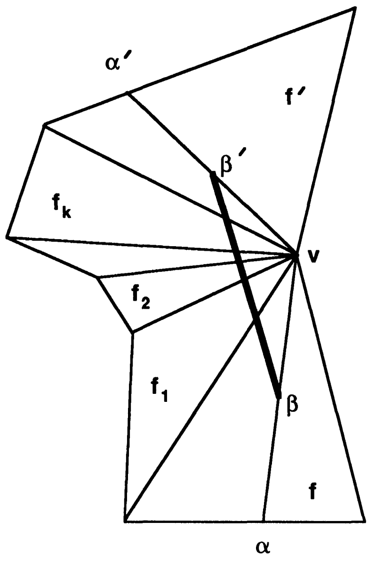

FIG. 4. *A shortcut for a path through a vertex.*

edge e which is contained in p). If p is geodesic, then by Lemma 3.3 the subpath from v to v' is completely determined by giving the edge sequence $\mathscr{E}$. Thus, we can describe a geodesic path p from s to x by writing it as a list, $p=(v_1=s, \mathscr{E}_1, v_2, \mathscr{E}_2, v_3, \cdots, v_K, \mathscr{E}_K, x)$, where $v_1, v_2, \cdots, v_K$ are the vertices (in order) through which p passes, and $\mathscr{E}_1, \mathscr{E}_2, \cdots, \mathscr{E}_K$ are (possibly empty) edge sequences which are connected by p. Note that $v_i=r(\mathscr{E}_i)$ for each i such that $\mathscr{E}_i \neq \emptyset$. We call v_K the *root of path* p and $\mathscr{E}_K$ the *last* edge sequence of p. For geodesic paths p, the edge sequences $\mathscr{E}_i$ may not be simple (e.g., recall the example of the string wound around the rectangular bar); however, if p is *optimal*, then each $\mathscr{E}_i$ *must* be simple and no edge is contained in more than one of the edge sequences $\mathscr{E}_i$, $1 \leqq i \leqq K$ (otherwise, p would enter a face more than once, violating Lemma 3.2).

In summary, we have the following characterization of geodesics and optimal paths on (possibly) nonconvex surfaces.

LEMMA 3.5. *The general form of a geodesic path is a path which goes through an alternating sequence of vertices and (possibly empty) edge sequences such that the unfolded image of the path along any edge sequence is a straight line segment and the angle of the path passing through a vertex is greater than or equal to π. The general form of an optimal path is the same as that of a geodesic path, except that no edge can appear in more than one edge sequence and each edge sequence must be simple.*

4. *f*-free paths and intervals of optimality.

For any point x, let $p(x)$ be an optimal path from s to point x and let $d(x)$ be the length of $p(x)$. The set of all points y on edge e such that there exists an optimal path to y with root r and last sequence $\mathscr{E}$ is some subset of e which can, in general, consist of up to $O(n)$ subsegments. This leads to a considerable complication when our algorithm tries to "propagate" signals from r across edge e. In the case of a convex polyhedron, this problem does not arise. In that case, the "wedges" in the planar unfolding of the surface are all convex since their boundaries are straight lines (which follows from the fact that optimal paths do not pass through vertices other than s, implying that bisectors between "wedges" are straight lines). In the nonconvex case, bisectors will be hyperbolic arcs, so the "wedges" will not be convex.

To overcome this difficulty, we define the notion of an *f-free path*: Given a face, f, and e, one of its three edges (we call (e,f) an *edge-face pair*), an *f-free path* to $y \in e$ is defined to be a path on $\mathscr{S}$ from s to y which does not intersect the interior of face f. A *shortest* f-free path to y, $p_f(y)$, is an f-free path to y which has minimal length. We will need to show the following.

LEMMA 4.1. *An optimal path $p(x)$ to a point $x \in e$ passes through the interior of at most one face containing x.*

Proof. If x is a vertex, there can be many faces containing x; if x is interior to e, there are exactly two. Let $f_1, f_2, \cdots, f_k$ be the faces containing x. If $p(x)$ passes through the interior of no face containing x, then we are done. Otherwise, let f_i be the *first* such face for which the optimal path, $p(x)$, has a point on its interior, and let y be a point on $p(x)$ which is interior to f_i. Then the subpath of $p(x)$ from y to x must be optimal. But the shortest path from y to x on $\mathscr{S}$ is just the straight segment from y to x, since this entire segment lies on face f_i and therefore on the surface $\mathscr{S}$. Thus, no other face $f_j \neq f_i$ containing x has the path $p(x)$ pass through its interior. □

Lemma 4.2 shows us how to obtain an optimal path to a point x on the interior of $e=f \cap f'$ from a shortest f-free and a shortest f'-free path to x.

LEMMA 4.2. *An optimal path to x on the interior of e is given by the shorter of $p_f(x)$ and $p_{f'}(x)$.*

Lemma 4.3 states another important property of shortest f-free paths: that they are not allowed to cross each other.

LEMMA 4.3. *If $p(x)$ and $p(y)$ are optimal paths from s to points x and y, then they can intersect only at vertices of $\mathcal{S}$, and if they do intersect at v, then that subpath of $p(x)$ from s to v has the same length as that subpath of $p(y)$ from s to v. A similar statement holds for shortest f-free paths $p_f(x)$ and $p_f(y)$.*

Proof. If $p(x)$ and $p(y)$ intersect at z, then certainly the subpaths to z must be equal in length (otherwise, one of the two paths could be shortened by replacing the portion of it from s to z). If z were interior to a face, then both paths could be locally improved by taking a shortcut around z. Similarly, if z were interior to an edge, a local improvement would be possible for both paths (look now at the unfolded paths and then improve them as if z were interior to a face). The same arguments hold for shortest f-free paths, since such local improvements will keep both paths f-free. □

Note that while $p_f(x)$ and $p_f(y)$ cannot cross, it is possible for shortest f-free paths to cross for *different* f's (e.g., $p_{f_1}(x)$ may cross $p_{f_2}(y)$; see Fig. 5). The problem is that if we try to improve one of the paths (say $p_{f_1}(x)$) by replacing part of it with a subpath of $p_{f_2}(y)$, then it may cease to be f_1-free.

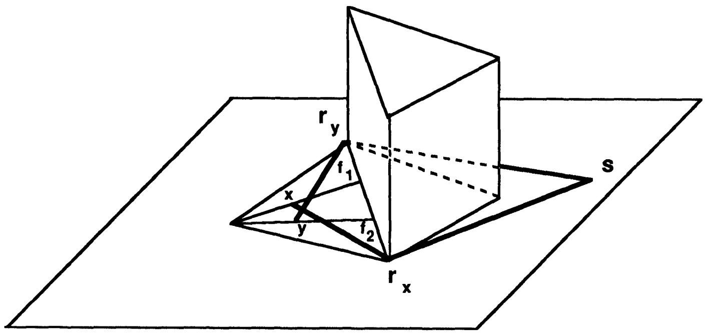

FIG. 5. *$p_{f_1}(x)$ may cross $p_{f_2}(y)$.*

The importance to our algorithm of the concept of shortest f-free paths is evident from the following lemma.

LEMMA 4.4. *For edge-face pair (e, f), edge sequence $\mathscr{E}$, and some vertex r, the set $\mathscr{I}$ of points x on e for which there exists a shortest f-free path to x with root r and last edge sequence $\mathscr{E}$ is connected (and, therefore, a subsegment of e).*

Proof. If there are no points of e which can be reached along a shortest f-free path with root r and last edge sequence $\mathscr{E}$, then we are done. Otherwise, let x and x' be points of e such that $p_f(x)$ and $p_f(x')$ have root r and last edge sequence $\mathscr{E}$. Then we wish to show that if y is a point of e between x and x', then $p_f(y)$ also has root r and last edge sequence $\mathscr{E}$. Now, unfold the edge sequence $\mathscr{E}$ and let $\bar{r} = U_{\mathscr{E}}(r)$ be the resulting unfolded image of r. Note that the segments $\overline{\bar{r}x}$, $\overline{\bar{r}x'}$, and $\overline{xx'}$ form a triangle in the unfolded image, since, by Lemma 3.3, the unfolded images of $\overline{rx}$ and $\overline{rx'}$ are straight line segments. Any f-free path, $p_f(y)$, to y cannot go through the interior of f; thus, $p_f(y)$ must pass through the interior of the triangle. Also, since shortest f-free paths do not cross (Lemma 4.3), the unfolded image of $p_f(y)$ cannot intersect the interiors of segments $\overline{\bar{r}x}$ and $\overline{\bar{r}x'}$. Thus, $p_f(y)$ must intersect the triangle at the point $\bar{r}$, meaning that $p_f(y)$ has root r and last edge sequence $\mathscr{E}$. □

The interval $\mathcal{I}$ of points can be open, closed, or half-open and half closed. (An open endpoint arises in the degenerate case in which the unfolding of the shortest f-free path to the point passes through two or more vertices (see Fig. 6). The fact that the resulting interval has an open endpoint is really an artifact of our definition of the root of a path as being the *first* vertex encountered when tracing back along the path.) Let the subsegment $I=[a, b]$ be the closure of $\mathcal{I}$. (Normally, we would write $I=\overline{ab}$ to indicate the segment with endpoints a and b; however, we write $[a, b]$ to emphasize the fact that coordinate values can be attached to points of e, and I can be thought of as an interval of these coordinates.) The endpoint a (resp., b) is the one on the *left* (resp., *right*) when viewed looking into face f. We call I the *interval of optimality* for r and $\mathscr{E}$ *with respect to* (e, f). An alternative specification of an interval of optimality is to give its *unfolded root* $\bar{r} = U_{\mathscr{E}}(r)$, its *depth* $d = d(r)$, and the corresponding edge-face pair (e, f). Note that r may be an endpoint of e (and hence of I) in the degenerate case. (Let us establish the convention that $\mathscr{E}$ does not include the edge e, so that $U_{\mathscr{E}}(r)$ writes the image of r in the coordinate system of face f'. If $\mathscr{E}=\varnothing$, then r will be one of the vertices of f'.) A point x is an element of the interval of optimality of $\bar{r}$ with respect to (e, f) if there exists a shortest f-free path to x whose unfolded image (along its last edge sequence) in the plane of f contains the segment $\overline{\bar{r}x}$.

We illustrate the notion of unfoldings and intervals of optimality in Fig. 6. On the left we show the interval I' of points on e which are "accessible" from r through the given edge sequence along paths that unfold into straight lines. Note that I' is determined by the left and right "clipping" vertices, r_L and r_R. (I' is simply that part of e which is visible from r within the polygon obtained by unfolding the sequence of adjacent faces.) In the figure on the right, we show the interval of optimality $I=[a, b]$. (Note also that in this case, $\mathcal{I}=(a, b]$ is half open, since the root of the

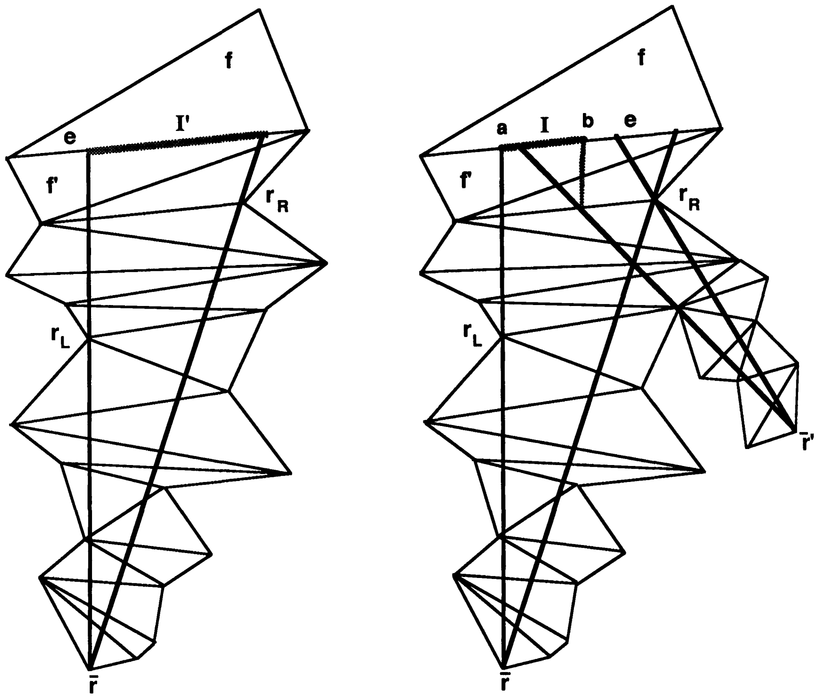

FIG. 6. *Unfoldings and intervals of optimality.*

path to a is r_L, not r.) The interval to the left of I is simply that which is rooted at r_L. The right endpoint, b, was determined by clipping I' along the hyperbolic bisector between $\bar{r}$ and $\bar{r}'$. Points right of b may be accessible from r, but they are better reached from r'.

It is easy to see that intervals of optimality with respect to (e, f) form a covering of e and that the intervals can touch but cannot overlap.

LEMMA 4.5. *Intervals of optimality with respect to an edge-face pair (e, f) form a covering of e and have mutually disjoint interiors.*

Proof. First note that every point $x \in e$ must lie in *some* interval of optimality with respect to (e, f) (if $p_f(x)$ is any shortest f-free path to x, then x lies in the interval of optimality of the unfolded image of the root of $p_f(x)$ with respect to (e, f)). Now, for a point $x \in e$ to be in the intervals of optimality of $\bar{r}$ and of $\bar{r}'$ with respect to (e, f) it must satisfy the equation $d(r) + |\bar{r}x| = d(r') + |\bar{r}'x|$. But this equation represents a hyperbola (which degenerates to a straight line if $d(r) = d(r')$). The intersection of this hyperbola with the edge e is a point (in the degenerate case, the straight line cannot contain e since the points $\bar{r}$ and $\bar{r}'$ would be on opposite sides of the line through e). Thus, there can be no open subinterval of e all of whose points lie in two or more intervals of optimality. This implies that the intervals must have disjoint interiors. □

The point, c, in I which is closest to r is called the *frontier point* of the interval. Since the distance from a point r to a closed line segment $[a, b]$ is minimized at exactly one point of $[a, b]$, we know that a unique such c exists. It will be either an endpoint of I or a point in the interior of I depending on whether the perpendicular projection of r onto the line through e lies outside or inside (respectively) the interval I.

The intersection of the f-free path to c with the face f' (which shares edge e with face f) is a line segment, $\overline{\beta c}$, on face f'. We call point β the *access point* of interval I. Note that when c is a vertex, it is possible for β to equal c, so that $\overline{\beta c}$ degenerates to a point. If we draw the line segments $\overline{\beta c}$ for every interval of optimality with respect to (e, f), then we get a partitioning of face f' into *access channels* (since no two such segments can intersect). Note that an access channel is either a triangle, a quadrilateral, or a pentagon. We will refer to an access channel by giving a pair, (I_1, I_2), where I_1 and I_2 are the intervals of optimality which give rise to the bordering segments, $\overline{\beta_1 c_1}$ and $\overline{\beta_2 c_2}$. If $I_1 = \text{NIL}$, the channel is simply that part of f' to the left of $\overline{\beta_2 c_2}$; and if $I_2 = \text{NIL}$, the channel is that to the right of $\overline{\beta_1 c_1}$.

5. The algorithm. Our algorithm operates in the precise spirit of Dijkstra's algorithm [1]. It is an important example of the power of the technique we are calling "continuous Dijkstra." A "signal" is propagated from the source to the rest of the surface. Once a point x of the surface receives the signal for the first time, it propagates it further; point x is considered to be *permanently labeled* with the time, $d(x)$, at which it received the signal, which is, of course, the minimum distance from the source to x. Fortunately, we have to do this labeling and repropagation for only finitely many (in fact, as we shall show, $O(n^2)$) points (called *event points*) of the surface. (Actually, we do a "directed" form of continuous Dijkstra, since we label points on an edge with the lengths of paths that are incident to the edge from either of the two sides of the edge. This comes from the fact that our intervals of optimality were defined with respect to f-free paths.)

The algorithm uses a few simple data structures. We keep a list, ILIST, of *candidate* intervals of optimality. A *candidate interval* (or *interval* for short) is a subsegment of an edge which is a supersegment of some (possibly empty) interval of optimality and has exactly the same structure as that interval (i.e., it has the same type of information associated with it in its data structure). We call them *candidate* intervals because, at the conclusion of our algorithm, all remaining candidate intervals will be intervals of optimality. A candidate interval, I, has the following information associated with it: its extent, $[a, b]$; its edge-face pair, (e, f); its root, r; its unfolded root, $\bar{r}$; its depth, d; its frontier point, c; its access point, β; and its predecessor, $\bar{I}$ (which is the candidate interval (or vertex) whose "propagation" originally gave rise to I). We then will write $I = ([a, b], (e, f), r, \bar{r}, d, c, \beta, \bar{I})$.

An edge-face pair, (e, f), has associated with it a sorted list of candidate intervals with respect to (e, f), with the intervals in the order that they appear along the edge (following the edge in the direction which keeps the face f on the *left*). Such an ordering is possible since candidate intervals will always have disjoint interiors. Initially and during the execution of the algorithm, the *interval list* of (e, f) need not *cover* the edge e (that is, there may be points of e that lie in no candidate interval), but at the conclusion of the algorithm, we will see that the list does form a covering.

Each vertex v has associated with it a pointer to each candidate interval of which it is a root. Additionally, there is a *permanent label*, $d(v)$, which indicates the shortest path length to vertex v. Initially, $d(v) = +\infty$ for every vertex v. As the algorithm proceeds, vertices will become *permanently labeled*, meaning that they are assigned a label $d(v) < +\infty$. We will also be labeling some nonvertex points (frontier points which are interior to edges). These are also assumed to have infinite initial labels.

The algorithm also maintains a log n priority queue (called the *event queue*) whose entries are points of some candidate interval (either an endpoint or the frontier point) and have labels which are the best-known distances back to the source. (We do not actually need to keep track of all of the endpoints of candidate intervals as event points. It would suffice to keep track only of events associated with vertices and with frontier points that are interior to their intervals; however, the efficiency of our algorithm is not affected by this change.)

The algorithm proceeds as follows.

ALGORITHM.

(0) (**Initialize**). Permanently label s with 0. Initialize the event queue and ILIST to be empty. For each edge-free pair, initialize its interval list to be empty. For each edge opposite to s, create a candidate interval whose extent is the entire edge and whose root is s. Compute c for each such interval and place each such c and the endpoints of the edges in the event queue, each point having as label its distance from s. Push the resulting intervals onto the list ILIST and onto interval lists of the corresponding edge-face pairs.

(1) (**Main Loop**). While there is an entry in the event queue, remove the one with the smallest label and permanently label it. If it is being labeled as the frontier point of some candidate interval, I, then do *Propagate* (I).

Intuitively, to *propagate* an interval I means to allow the "wave front" of signals from the root of I to pass through I to the other two edges of the face and to record on those edges the intervals of points that are optimally reached by the signal paths through I. One interval can propagate into two (one on each of the opposite edges), so we will have to be careful how we propagate intervals so that the number of them does not explode exponentially. The critical observation which allows for a "trimming" of the number of intervals is that optimal paths and shortest f-free paths *do not cross* (Lemma 4.3).

Thus, the propagation of intervals can take place only through the access channels already established by intervals on the opposite edges. We give now the details of the procedure *Propagate* (I):

PROCEDURE *Propagate* (I).

(0) Assume that I is an interval with respect to (e, f), where $e = f \cap f'$. Let $c \in e$ be the frontier point of I. Let $e_1 = f \cap f_1$ and $e_2 = f \cap f_2$ be the edges of f opposite edge e. Refer to Fig. 7.

(1) (c **is** not **a vertex**). Let I_i = *Project* (I, e_i) and, if $I_i \neq$ NIL, do *Insert-Interval* (I_i, c), for $i = 1, 2$.

(2) (c **is a vertex**). For each edge e_0 opposite c such that some point of e_0 makes an angle greater than π with segment $\overline{\beta c}$, create a candidate interval, I_0, on e_0 whose extent is the entire edge e_0 and whose root is c. Do *Insert-Interval* (I_0, c).

Step (2) makes use of Lemma 3.4 to specify when it is reasonable to introduce vertex c as a root to a candidate interval on an opposite edge. (This is not essential to the correctness of the algorithm or to its complexity, but only serves to make it run slightly faster.) We only introduce intervals on edges for which there is a chance that the angle that the resulting paths will make at c will be greater than π.

The function *Project* (I, e_1) simply finds the subset of e_1 that is hit by paths through I. We just apply the local optimality criterion (that paths unfold into straight lines; Lemma 3.3) to determine how the "wedge" of paths is extended through the next face. The function returns a candidate interval, I_A, on e_1 whose points are accessible through I. (If no part of e_1 is accessible, we return NIL.) The root of I_A is the same as that of I, and the unfolded root of I_A is obtained from that of I simply by rotating it (about e) into the coordinate system of face f'.

The procedure *Insert-Interval* (I, c) simply inserts a new candidate interval, I, between the intervals I_1 and I_2 that define the access channel containing c. The interval I may "dominate" I_1 or I_2 (which can happen only if the corresponding frontier point of I_1 or I_2 has not yet been permanently labeled), in which case the dominated candidate

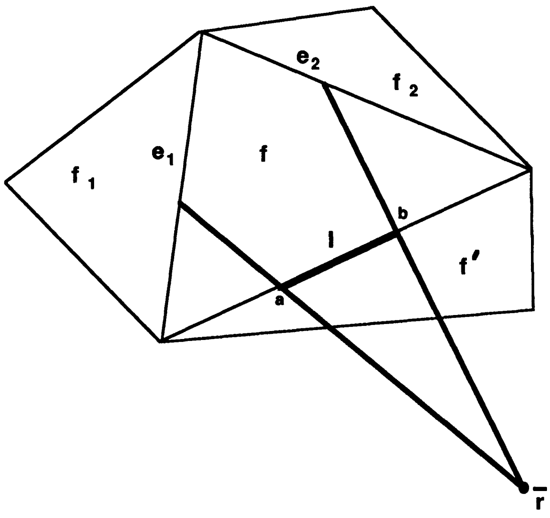

FIG. 7. *Propagation of an interval.*

must be deleted. Also, I will possibly be "trimmed" at the "tie points" between the root of I and the roots of I_1 and I_2. Thus, the portion of I which survives the trimming will correspond to that part of the edge containing I which will be "hit" first by signals from the root of I which pass through the last edge sequence of I.

PROCEDURE *Insert-Interval* (I, c).

(0) Assume $I=[a, b]$ is an interval with respect to (e,f) and that it has root r, unfolded root $\bar{r}$, and depth d. Point c is the frontier point of $\bar{I}$, the predecessor of I.

(1) (**Locate point c in access channel**). Locate point c in an access channel, say (I_1, I_2), of (e,f). The wavefront can only get to I through this channel. Let $I_i=[a_i, b_i]$ (provided that $I_i \neq$ NIL), for $i=1,2$. Let r_i be the root, $\bar{r}_i$ be the unfolded root, and d_i be the depth of I_i.

(2) (**Delete dominated candidate intervals**). If $I_1 \neq$ NIL, $a_1 \in I$, and $d_1+|\bar{r}_1 a_1| \geqq d+|\bar{r} a_1|$ then do *Delete* (I_1), set I_1 to be the predecessor of I_1 in the interval list of e_1 (set I_1 to NIL if no predecessor exists), and go back to (2). Similarly, if $I_2 \neq$ NIL, $b_2 \in I$, and $d_2+|\bar{r}_2 a_2| \geqq d+|\bar{r} a_2|$, then do *Delete* (I_2), set I_2 to be the successor of I_2 in the interval list of e_1 (set I_2 to NIL if no successor exists), and go back to (2).

(3) (**Trim at tie points**). If $I_1 \neq$ NIL and $b_1 \in I$, then set a to be the point in $I \cap I_1$ such that $d_1+|\bar{r}_1 a|=d+|\bar{r} a|$ (if no such point exists, let a be b_1). Then a will be the point in $I \cap I_1$ which can be reached equally well through I and through I_1. Remove b_1 from the event queue, set b_1 to a, and update c_1 if necessary. If $I_2 \neq$ NIL and $a_2 \in I$, then set b to be the point in $I \cap I_2$ such that $d_2+|\bar{r}_2 b|=d+|\bar{r} b|$ (if no such point exists, let b be a_2). Then b will be the point in $I \cap I_2$ which can be reached equally well through I and through I_2. Remove a_2 from the event queue, set a_2 to b, and update c_2 if necessary. If $a \neq b$, go to (4); otherwise, stop (no interval needs to be inserted).

(4) (**Final updates**). Compute the frontier point, c_I, and access point for $I=[a, b]$. Push I onto the list ILIST and insert it into the interval list of edge e (between intervals I_1 and I_2). Put the points a, b, and c_I (if $c_I \neq a, b$) into the event queue, with labels $d+|\bar{r} a|$, $d+|\bar{r} b|$, and $d+|\bar{r} c_I|$, respectively.

The procedure *Delete* (I) simply deletes I from the ILIST, deletes I from the interval list of the corresponding edge-face pair, and deletes any entries of points of I in the event queue.

In step (1) of the above procedure, we must locate a point c in an access channel of an edge-face pair (e,f). This is easily done in time $O(\log n)$ by binary search as follows: We have for (e,f) an interval list which gives the segments $\overline{\beta_i c_i}$ in sorted order along edge e. The segments are nonintersecting, so to locate a point c on face f' in an access channel, we can do binary search in which each test determines on what side of a segment $\overline{\beta_i c_i}$ the point c lies (see Fig. 8).

6. Proof of correctness. We now begin a proof that the algorithm works. Our first objective is to prove the following proposition.

PROPOSITION 6.1. *After each iteration of the Main Loop of the Algorithm, the current list of candidate intervals (ILIST) correctly gives the best f-free path so far from s to points of these intervals, where "best so far" has the following meaning: If $x \in e=f \cap f'$ lies in interval I of root r with respect to (e,f), then an f-free path to x which has root r and length $d(r)+|\bar{r}x|$ is optimal among those f-free paths to x that enter face f' through an interval in the current ILIST.*

Proof. The proof proceeds by induction on the iteration count in the Main Loop. At the first iteration, the claim is true because the only candidate intervals are the trivial ones on the edges opposite to the source s. Assume now that the claim holds for the first k iterations (this is the Induction Hypothesis). The inductive step requires

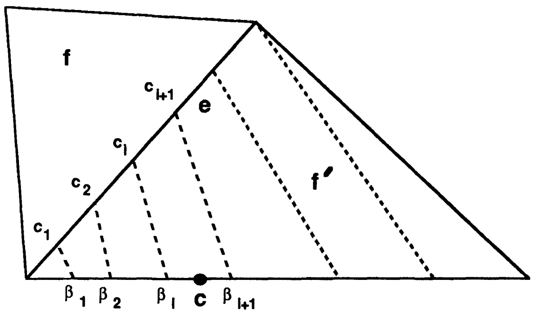

FIG. 8. *Locating point c in a channel.*

us to prove that if, at iteration $k+1$, we introduce interval I for root r with respect to (e,f), then the best f-free path so far to points of I is through root r. This will be shown in the following two lemmas. □

Assume now that the Induction Hypothesis holds.

A candidate interval will have one of two fates: Either its frontier point gets permanently labeled (by its becoming the top element in the priority queue), thereby assuring that some part of it will "survive" and become part of a final interval of optimality with the same attributes as the candidate; or, the candidate interval gets deleted in Step (2) of *Insert-Interval*, in which case we can guarantee that the interval of optimality corresponding to the candidate will be empty.

LEMMA 6.2. *If we delete interval I_1 in step* (2) *of procedure Insert-Interval, then the interval of optimality of root r_1 with respect to (e,f) has an empty interior. A similar statement holds if we delete interval I_2.*

Proof. Assume that $a_1 \in I$ and $d_1 + |\bar{r}_1 a_1| \geqq d + |\bar{r} a_1|$. Then, the best f-free path so far to point a_1 is through root r. Let β be the intersection of $\overline{\bar{r}a_1}$ with edge e'. Since $a_1 \in I$, we know that $\beta \in \bar{I}$, by the method used to construct I from its predecessor, $\bar{I}$. Let us assume that there exists a point y in the interior of the candidate interval of root r_1 with respect to (e,f) (so the path $p_f(y)$ has root r_1). We will then arrive at a contradiction.

First, note that y is interior to $[a_1, b_1]$ (otherwise, y is either inaccessible from r_1 or can be reached from some other root of (e,f) along a shorter f-free path). Next, note that interval I_1 was established when its predecessor, $\bar{I}_1$, was propagated. Since point $c \in \bar{I}$ was located in channel $(\bar{I}_1, \bar{I}_2)$, we know that c lies to the *right* of the segment $\overline{\bar{r}_1 c_1}$, and therefore to the right of $\overline{\bar{r}_1 b_1}$ (otherwise, $c \in \bar{I}_1$). Since $\bar{I}$ is connected, then, *all* points of $\bar{I}$ lie to the right of $\overline{\bar{r}_1 z}$ for *any* $z \in [a_1, b_1]$, and lie *strictly* to the right of $\overline{\bar{r}_1 z}$ for z *interior* to $[a_1, b_1]$. In particular, β lies strictly to the right of the segment $\overline{\beta_1 y}$, where $\overline{\beta_1 y}$ is the intersection of $\overline{\bar{r}_1 y}$ with face f'. But, a_1 lies strictly to the *left* of segment $\overline{\beta_1 y}$, so segments $\overline{\beta a_1}$ and $\overline{\beta_1 y}$ must intersect in some point γ which is interior to both segments (see Fig. 9). Since γ lies on the path $p_f(y)$, the f-free path through r_1 to γ is no longer than the f-free path through r to γ. But this means that we can get a strict improvement to the path to a_1 by going through root r_1 directly to a_1 (since $|\bar{r}_1\gamma| + |\gamma a_1| > |\bar{r}_1 a_1|$). This contradicts our assumption that $d_1 + |\bar{r}_1 a_1| \geqq d + |\bar{r} a_1|$.

A similar proof holds if we delete interval I_2. □

Next, we must prove that the interval that is inserted by procedure *Insert-Interval* is "correct" in the following sense.

LEMMA 6.3. *When procedure Insert-Interval inserts an interval I between intervals I_1 and I_2, the resulting interval list of edge e correctly subdivides points of e according to the best f-free paths so far.*

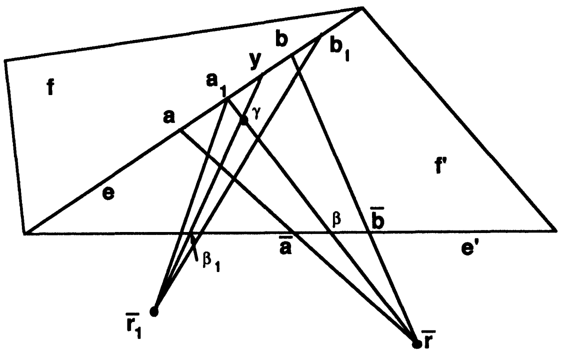

FIG. 9. *Illustration of proof of Lemma* 6.2.

Proof. Let $[a_1, b_1]$ and $[a_2, b_2]$ be the original extents of I_1 and I_2, and let $[a_1, b_1']$ and $[a_2', b_2]$ be the resulting intervals after the trimming is done by procedure *Insert-Interval.* Now, the best path so far to a point outside $[a_1, b_2]$ cannot go through r since it would have to cross some access channel boundary, $\overline{\beta_i c_i}$.

Now let $x \in I = [a, b]$, where $[a, b]$ is the extent of I *after* any trimming that may have been done. Since, by construction, $I \subseteq [a_1, b_2]$, the best path so far to x must be through r, r_1, or r_2. If x is not in $[a_1, b_1]$, then it is better to reach x through root r than through root r_1. If x is interior to $[a_1, b_1]$, then so is a (since $[a, b] \subseteq [a_1, b_2]$). But then the path to a through root r is no longer than the path to a through root r_1 and is, therefore, optimal so far. If it were better to reach x through root r_1 than through root r, then the path to x would intersect the path through r to a (since segments $\overline{\bar{r}a}$ and $\overline{\bar{r}_1 x}$ must cross), a contradiction. Likewise, we can show that it is better to reach x through r than through r_2.

If $x \in [a_1, b_1']$ (resp., $[a_2', b_2]$), a similar argument shows that it is better to reach x through r_1 (resp., r_2) than through r (and it is certainly better to reach x through r_1 (resp., r_2) than through r_2 (resp., r_1) since $[a_1, b_1'] \subseteq [a_1, b_1]$ and $[a_2', b_2] \subseteq [a_2, b_2]$). □

This completes the proof of Proposition 6.1.

One can show that during an execution of Dijkstra's algorithm, when a node is "permanently labeled" (i.e., put on the "CLOSED" list), it is labeled with the length of the shortest path from the source to the node. Analogously, we have the following lemma.

LEMMA 6.4. *When a point interior to an edge is permanently labeled as part of a candidate interval with respect to (e, f), it is labeled with the shortest f-free path length to it. When a vertex is permanently labeled, it is labeled with the shortest path length to it.*

Proof. Let x be interior to edge $e = f \cap f'$. Assume that the algorithm has just given x a permanent label of δ and that the path $p_f(x)$ is a shortest f-free path to x whose length is less than δ. Now, $p_f(x)$ intersects at least one candidate interval (since we initialized the ILIST to "surround" point s), so we can let $I_l = [a_l, b_l]$ be the *last* candidate interval intersected by $p_f(x)$, and assume that I_l is an interval with respect to (e_l, f_l). (Note that there will be more than one such interval if $p_f(x)$ intersects I_l at one of its endpoints; in that case, pick either interval. In the case that $p_f(x) \cap I_l = v$, a vertex, pick I_l such that $p_f(x) \cap f_l \neq v$.) Since the length of $p_f(x)$ is less than δ, we know that $p_f(x)$ enters face f' through some point which is in no current candidate interval (by Proposition 6.1, since x was labeled with the best path length so far). Thus, $f_l \neq f'$. Let $\overline{\beta x'} = f_l \cap p_f(x)$. Then, point x' lies on an edge $e_0 = f_l \cap f_0$. Let $p_{f_0}(x')$ be the subpath of $p_f(x)$ from s to x'. Clearly, $p_{f_0}(x')$ is a shortest f_0-free path to x'.

The interval I_l *must* have been propagated (since the distance to the frontier point of I_l is $\leqq d(\beta) < d(x) < \delta$, and the algorithm always permanently labels the closest event point). But then, by Lemma 6.3, there would have been a candidate interval created on edge e_0 which includes point x'. This contradicts the fact that I is the *last* candidate interval intersected by $p_f(x)$.

A similar argument shows that vertices are permanently labeled with the length of an optimal path to them. □

In our mental picture of the execution of the algorithm, we think of the "wavefront" moving out from s and imagine that, as the front encounters new edges or boundaries of candidate intervals, events occur. When an event occurs at a certain distance from s, then all points closer to s than that distance have already encountered the "wavefront," as Lemma 6.5 states.

LEMMA 6.5. *If the algorithm has just assigned a permanent label of δ to some point, then for any point x on an edge e such that $d(x) < \delta$, there is an interval I (for r with respect to (e, f)) in the current ILIST which contains x such that an optimal path to x has root r and length $d(r) + |\bar{r}x|$.*

Proof. Let $x \in e$ be an edge point such that $d(x) < \delta$. Assume that x does *not* lie in a candidate interval which correctly identifies an optimal path to x. Let $p(x)$ be an optimal path to x. As in the proof of Lemma 6.4, let $I_l = [a_l, b_l]$ be the *last* candidate interval of the current ILIST intersected by $p(x)$, and assume that I_l is an interval with respect to (e_l, f_l). Our assumption on x says that $x \notin I_l$. Let $\overline{\beta x'} = f_l \cap p(x)$. Then, point x' lies on an edge $e_0 = f_l \cap f_0$. Let $p_{f_0}(x')$ be the subpath of $p(x)$ from s to x'. Clearly, $p_{f_0}(x')$ is a shortest f_0-free path to x'. The interval I_l *must* have been propagated (since the distance to the frontier point of I_l is $\leqq d(\beta) < d(x) < \delta$, and the algorithm always permanently labels the closest event point). But then, by Lemma 6.3, there would have been a candidate interval created on edge e_0 which includes point x'. This contradicts the fact that I_l is the *last* candidate interval intersected by $p(x)$. □

At the conclusion of our algorithm, the wavefront has encountered *all* points of $\mathcal{S}$:

LEMMA 6.6. *At the conclusion of the algorithm, the interval list of (e, f) forms a covering of the edge e.*

Proof. Assume that there is some point $x \in e = f \cap f'$ which is *not* covered by a candidate interval with respect to (e, f). Let $p_f(x)$ be a shortest f-free path to x. As in the proof of Lemma 6.4, let $I_l = [a_l, b_l]$ be the *last* candidate interval intersected by $p_f(x)$, and assume that I_l is an interval with respect to (e_l, f_l). Since x is not covered, $e_l \neq e$. Let $\overline{\beta x'} = f_l \cap p_f(x)$. Then, point x' lies on an edge $e_0 = f_l \cap f_0$. Let $p_{f_0}(x')$ be the subpath of $p_f(x)$ from s to x'. Clearly, $p_{f_0}(x')$ is a shortest f_0-free path to x'. The interval I_l *must* have been propagated (otherwise, its frontier point would still be in the event queue, and the algorithm would not have concluded). But then, by Lemma 6.3, there would have been a candidate interval created on edge e_0 which includes point x'. This contradicts the fact that I_l is the *last* candidate interval intersected by $p_f(x)$. □

LEMMA 6.7. *The following are equivalent for a point x on edge e of face f.*

(1) *At the conclusion of the algorithm, x lies in the candidate interval of root r (and unfolded root $\bar{r}$) with respect to (e, f).*

(2) *There is a shortest f-free path p from s to x with root r and length $d(r) + |\bar{r}x|$; that is, x lies in the interval of optimality of $\bar{r}$ with respect to (e, f).*

Proof. The proof follows immediately from Proposition 6.1 and Lemma 6.6. □

7. Complexity analysis. To establish the running time of our algorithm, we first must bound the number of event points.

LEMMA 7.1. *There are at most $O(n^2)$ candidate intervals created by the algorithm.*

Proof. Let $I_1, I_2, \cdots, I_K$ be the interval list for some edge-face pair (e, f). Let x_i be a point interior to interval I_i, and let edge $e = \overline{x_0 x_{K+1}}$. Now draw the shortest f-free paths from s to each x_i $(0 \leqq i \leqq K+1)$. (Note that none of these paths cross edge e.) It is clear that this divides the polyhedral surface $\mathcal{S}$ into $K+1$ subsurfaces (subsurface i is the region to the right of the cycle from s to x_i (along $p_f(x_i)$), from x_i to x_{i+1} (along e), and then from x_{i+1} back to s (along the reverse of $p_f(x_{i+1})$). Note that paths $p_f(x_i)$ and $p_f(x_{i+1})$ may have some vertices in common. The interior of each such subsurface must contain a vertex. Thus, $K \leqq n-1$. Since there are at most $O(n)$ edge-face pairs, there can be at most $O(n^2)$ candidate intervals.

The above argument applies only to the case of a polyhedron of genus zero, as we are implicitly using the Jordan curve theorem to say that the surface is partitioned into $K+1$ subsurfaces. The case in which the genus may be higher can be handled by the following alternative proof. We trace back the paths from x_i and from x_{i+1}, noting where the paths first diverge by going through different edges (or where one path encounters a vertex, its root). Define the *fork point*, v_i, to be the vertex responsible for the divergence. It may be a root of one of the paths, or it may be a vertex between two adjacent edges whose interiors are crossed by the two paths. Let f_i be the face on which the divergence occurred; that is, both $p_f(x_i)$ and $p_f(x_{i+1})$ pass through the interior of f_i, and v_i is the vertex of f_i which "splits" paths $p_f(x_i)$ and $p_f(x_{i+1})$ (and it may lie on one of the paths). Then, we "charge" the interval I_i to the vertex-face pair, (v_i, f_i). It is not hard to see that each vertex-face pair can be charged at most twice: If v_i is a root of one path (say $p_f(x_i)$), then (v_i, f_i) can be charged at most twice (once for the pair $p_f(x_{i-1})$, $p_f(x_i)$, and once for the pair $p_f(x_i)$, $p_f(x_{i+1})$); otherwise, (v_i, f_i) is charged at most once. (The proofs of these claims simply use the fact that shortest f-free paths do not cross.) There is a one-to-one mapping between vertex-face pairs and edges, so we get that each edge can be charged at most twice. This gives us a linear bound for K in terms of the number of *edges* of $\mathcal{S}$ (which seems to be the proper measure of complexity for higher genus polyhedra). □

We can also bound the number of calls to *Delete* in procedure *Insert Interval.*

LEMMA 7.2. *There will be at most $O(n^2)$ calls to procedure Delete.*

Proof. By Lemma 6.2, we never delete an interval which has had some point permanently labeled. Once an interval is deleted, it can never be propagated. When an interval is propagated, its frontier point is permanently labeled and there are at most two new intervals created on the opposite edges. These two new intervals are candidate intervals which may end up being deleted. But there can be at most $O(n^2)$ such candidates, since, by Lemma 7.1, there can be at most $O(n^2)$ intervals with permanently labeled points. □

The size of the data structures needed is easily seen to be quadratic.

LEMMA 7.3. *The data structures require at most $O(n^2)$ space.*

Proof. Each of the $O(n^2)$ candidate intervals requires only a constant amount of storage. (Note that we do *not* keep the edge sequence corresponding to each interval, rather only the image of the root.) Since each edge is adjacent to only two faces, there are only $O(n)$ edge-face pairs, and each of these has associated with it a list of at most $O(n)$ candidate intervals (by the proof of Lemma 7.1). Thus, the data structure for the edge-face pairs requires only $O(n^2)$ space. The event queue can have at most three times the number of candidate intervals, so this structure is also bounded in size by $O(n^2)$. □

We are finally ready to assert the validity of our algorithm and its claimed efficiency.

THEOREM 7.4. *The algorithm above correctly computes the subdivision of each edge into intervals of optimality in time $O(n^2 \log n)$ and space $O(n^2)$.*

Proof. Correctness is based on Lemma 6.7. The space bound is shown in Lemma 7.3. For the time bound, we know from Lemma 7.1 that there are at most $O(n^2)$ events placed in the event queue. Since each event will be inserted and deleted only once, the total time needed for operations on the queue is $O(n^2 \log n)$. Also, by Lemma 7.1, there will be $O(n^2)$ calls to the procedure *Propagate.* If the frontier point being labeled is *not* a vertex, *Propagate* takes time $O(\log n)$ (just the time to locate c in an access channel) and then calls *Insert-Interval.* Each cell to *Insert-Interval* requires only constant time *except* for the work needed to call *Delete.* But each call to *Delete* requires at most $O(\log n)$ time (to delete an element from an interval list), so, by Lemma 7.2, the total time required is $O(n^2 \log n)$. In the case that the frontier point *is* a vertex, a single call to *Propagate* could potentially take time $O(n \log n)$ (since there could be $O(n)$ faces adjacent to c and each call to *Insert-Interval* takes $O(\log n)$ time). But the *total* number of calls to the procedure *Insert-Interval* cannot exceed the number of pairs (v, e), where v is a vertex, and e is the edge opposite v. Clearly, there can be at most $O(n)$ such pairs. Thus, the total time required in making calls to *Insert-Interval* is at most $O(n^2 \log n)$. This gives a total time bound of $O(n^2 \log n)$. □

8. Building the subdivision. The algorithm we presented allows us to recover the length, $d(x)$, of the shortest path from s to any point x on any edge $e = f \cap f'$. If x is a vertex, then $d(x)$ is just the label that was given it when x was permanently labeled. Otherwise, we locate x (in time $O(\log n)$) in interval of optimality I (resp., I') for root r (resp., r') with respect to (e, f) (resp., (e, f')). By Lemma 4.2, the smaller of the two sums, $d(r)+|x\bar{r}|$ and $d(r')+|x\bar{r}'|$ is the desired shortest path length.

We can extend our algorithm to find the shortest path length from s to any point of the polyhedral surface, as follows: Let e_1, e_2, and e_3 be the three edges of f. We build three subdivisions ($\mathcal{T}_1$, $\mathcal{T}_2$, and $\mathcal{T}_3$) of face f. Subdivision $\mathcal{T}_i$ is the planar subdivision of f into cells associated with the intervals of optimality with respect to (e_i, f).

Let $I_1, I_2, \cdots, I_K$ be the interval list for edge-face pair (e_1, f). Associated with each interval of optimality I_j $(1 \leqq j \leqq K)$ is a cell C_j of $\mathcal{T}_1$ such that a point x lies in cell C_j if and only if there is a path (on $\mathcal{S}$) from s to x through interval I_j (that is, through the root of I_j and its edge sequence) which is shortest among all paths (on $\mathcal{S}$) from s to x which pass through edge e_1.

Let interval I_j have root r_j. Then a point x will be on the boundary of cells C_j and $C_{j'}$ if and only if $d(r_j)+|x\bar{r}_j| = d(r_{j'})+|x\bar{r}_{j'}|$, where $\bar{r}_j = U_{(\mathscr{E}_j, e)}(r_j)$ is the image of r_j unfolded into the plane of f (this is slightly different from the convention we have had so far that $\bar{r} = U_{\mathscr{E}}(r)$ is the unfolded image of r into the plane on the other side of e_1). Put another way, the *weighted* distance from x to $\bar{r}_j$ must equal that from x to $\bar{r}_{j'}$, where by weighted distance between a point x and another point y (which has an associated *weight* d) we mean the sum $d+|xy|$. From elementary analytic geometry, we know that the locus of points whose weighted distances to two points r and r' are equal is a hyperbola. Thus, the boundaries of cells C_j in the subdivision are hyperbolic arcs.

We now describe a procedure, *Build-Subdivision* (e, f), which builds the subdivision of f into cells C_j which give the best paths through e to points x on f. The procedure works by "propagating" the intervals of optimality $(I_1, I_2, \cdots, I_K)$ with respect to (e, f) into the interior of face f, keeping track of certain "event" points (which will be crossings of hyperbolic arcs of cell boundaries).

We let $h_{i,j}$ be the hyperbola containing the boundary between cell C_i (associated with interval $I_i = [a_i, b_i]$) and cell C_j (associated with interval $I_j = [a_j, b_j]$). Then $h_{i,j}$

is defined by the equation $d(r_i)+|x\bar{r}_i| = d(r_j)+|x\bar{r}_j|$ in x (in the coordinate system of face f). Note that $h_{i,j}$ is nonempty provided that $|d(r_i)-d(r_j)| \leqq |\bar{r}_i\bar{r}_j|$, where the case of equality yields a hyperbola which degenerates to a line. We truncate $h_{i,j}$ to be in the half-plane defined by the line through e and containing face f. We then think of $h_{i,j}$ being *oriented* in the direction of increasing distance from r_i and r_j. We define $h_{0,1}$ as the ray from a_1 along edge e_3 and $h_{K,K+1}$ as the ray from b_K along edge e_2. If cells C_i and C_j share a boundary, then we let $q_{i,j}$ $(0 \leqq i < j \leqq K+1)$ be the point of the boundary of C_i and C_j which is closest to point r_i (and r_j). See Fig. 10 for an example in which $K=6$. Finally, let u_i be the supremum of the distances from s to points of C_i (note that u_i may be $+\infty$).

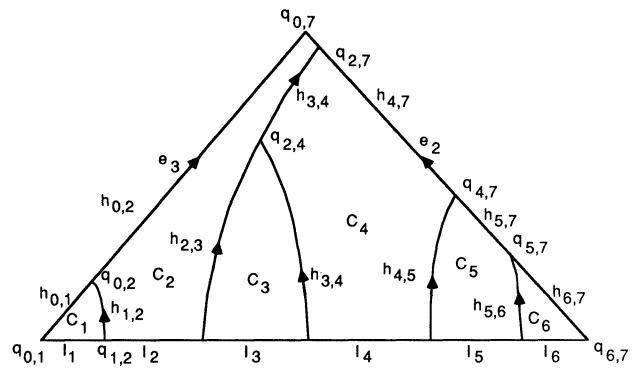

FIG. 10. *Extending the subdivision to the interior of f.*

PROCEDURE *Build-Subdivision* (e, f).

(0) Initialize INTLIST to be the list $(I_0, I_1, I_2, \cdots, I_K, I_{K+1})$. For each i $(1 \leqq i \leqq K)$, let $u_i = d_i + |\bar{r}_i w_i|$, where w_i is the point of intersection of $h_{i-1,i}$ and $h_{i,i+1}$ if they intersect, and let $u_i = +\infty$ if they do not intersect. Maintain the u_i's in a priority queue, ULIST.

(1) (**Determine Next Event**). If ULIST is empty (meaning that INTLIST equals (I_0, I_{K+1})), then stop. Otherwise, let the elements of INTLIST be indexed $(I_0, I_{i_1}, \cdots, I_{i_J}, I_{K+1})$. Let $u = u_{i_j}$ be the minimum value in ULIST (in case of ties, pick the one with smallest index i_j). Delete I_{i_j} from INTLIST, and delete u_{i_j} from ULIST.

(2) (**Update $u_{i_{j-1}}$ and $u_{i_{j+1}}$**). If $j \neq 1$, modify $u_{i_{j-1}}$ to be $d_{i_{j-1}} + |\bar{r}_{i_{j-1}} w_{j-1}|$, where w_{j-1} is the point of intersection of $h_{i_{j-2}, i_{j-1}}$ and $h_{i_{j-1}, i_{j+1}}$ if they intersect, and set $u_{i_{j-1}} = +\infty$ if they do not intersect. If $j \neq J$, modify $u_{i_{j+1}}$ to be $d_{i_{j+1}} + |\bar{r}_{i_{j+1}} w_{j+1}|$, where w_{j+1} is the point of intersection of $h_{i_{j-1}, i_{j+1}}$ and $h_{i_{j+1}, i_{j+2}}$ if they intersect, and set $u_{i_{j+1}} = +\infty$ if they do not intersect. Bisectors h_{i_{j-1}, i_j} and $h_{i_j, i_{j+1}}$ become "inactive," and bisector $h_{i_{j-1}, i_{j+1}}$ becomes "active" at this point.

(3) (**Update Subdivision**). Let $q_{i_{j-1}, i_{j+1}}$ be the event point which corresponds to u_{i_j}. (Point $q_{i_{j-1}, i_{j+1}}$ is the point of C_{i_j} which is furthest from r_{i_j}.) Establish $q_{i_{j-1}, i_{j+1}}$ as a vertex of the subdivision, with incident arcs h_{i_{j-1}, i_j} and $h_{i_j, i_{j+1}}$ (into $q_{i_{j-1}, i_{j+1}}$) and $h_{i_{j-1}, i_{j+1}}$ (out of $q_{i_{j-1}, i_{j+1}}$). Go to step (1).

The procedure works by processing the vertices of $\mathcal{T}_1$ in the order of increasing distance from the source. Each event corresponds to "closing off" a cell C_{i_j}, at which time we delete I_{i_j} from the list INTLIST of intervals responsible for the wavefront, and we consider the effect of the bounding hyperbola between the cells corresponding to the intervals on either side of I_{i_j} in list INTLIST. A typical step of the procedure is illustrated in Fig. 11. The lemma below shows that the procedure *Build-Subdivision* (e, f) does what it claims to do.

LEMMA 8.1. *The procedure Build-Subdivision (e, f) correctly constructs the cells C_j corresponding to the intervals of optimality of (e, f).*

Proof. The proof is by induction on the iteration count of the main loop. Assume that δ_k is the distance of the event point from s at iteration k. We first note that the subdivision induced by the initial set of hyperbolic bisectors is correct up to a distance δ_1 from s (since, up to distance δ_1, there are no intersections among the bisectors, by the definition of the event distance). Now assume that it is correct up to distance δ_k, and consider the effect of the event at distance δ_k. Event k corresponds to the closing off of a cell C_I and to the cells C_L and C_R becoming adjacent after distance δ_k. (C_L and C_R had been to the left and to the right, respectively, of C_I before the event.) If x is a point at distance δ from s, where $\delta_k \leqq \delta < \delta_{k+1}$, then we claim that x is properly categorized according to the current list of active hyperbolic bisectors, that is, according to the subdivision created by the procedure for points between event k and event $k+1$. First, x cannot be reached best through interval I, since it lies on the wrong side of the bisector between C_I and C_L or between C_I and C_R. The newly active bisector between C_L and C_R correctly categorizes x between the corresponding intervals. And the categorization of x with respect to all other intervals must be correct since all other bisectors are disjoint for distances from s which are between δ_k and δ_{k+1}, and by the induction all other bisectors are disjoint for distances from s which are between δ_k and δ_{k+1}, and by the induction hypothesis, we have assumed that the subdivision is correct so far. □

LEMMA 8.2. *The procedure Build-Subdivision (e, f) requires time $O(K \log K)$ and space $O(K)$, where K is the number of intervals of optimality with respect to (e, f).*

Proof. Since at each iteration of the procedure we delete one of the intervals from INTLIST, there can be at most K executions of steps (1) and (2). But each such

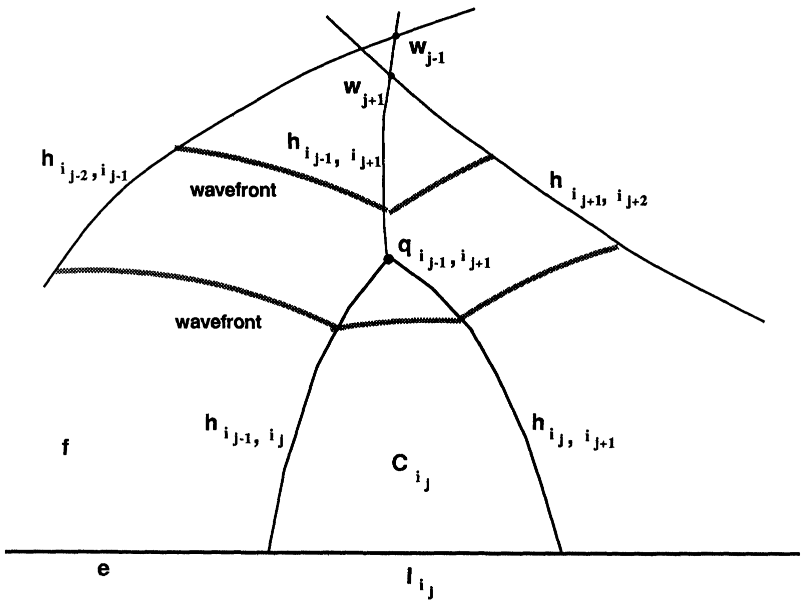

FIG. 11. *The event at point $q_{i_{j-1}, i_{j+1}}$.*

execution involves a constant number of insertions and deletions from the list ULIST. Thus, the total time complexity of the procedure is clearly $O(K \log K)$. □

We should remark that it is not necessary to build independently each of the three subdivisions $\mathcal{T}_1$, $\mathcal{T}_2$, and $\mathcal{T}_3$. (This was done for simplicity of exposition.) Rather, we could have propagated the intervals for (e_1, f), (e_2, f), and (e_3, f) all at once into the interior of f. The procedure would be similar to that given above for a single edge, but now the INTLIST would be a cyclic list of intervals around the perimeter of f (eliminating the need for the "dummy" hyperbolas on the edges of f).

We execute the procedure *Build-Subdivision* (e, f) for every edge-face pair (e, f). This yields three subdivisions for each face and takes a total time of $O(n^2 \log n)$ (since there are a total of $O(n^2)$ intervals of optimality for all edge-face pairs). To find the length of the shortest path from s to any $x \in S$ we simply locate x in some face f, then we do three point locations of x within the three subdivisions of f. We now have the lengths of shortest paths to x through each of the three edges of f. Selecting the shortest of the three gives us the length of the shortest path to x. All four of the point locations can be done in time $O(\log n)$ by standard point location techniques ([15], [3]). (The only slight variation to the problem with which we must deal is the fact that the boundaries of cells are hyperbolic instead of straight. This, however, presents no real complication.) To determine an actual optimal path to x is straightforward by back-tracing through the predecessor pointers (from I to $\bar{I}$ to $\bar{\bar{I}}$, etc.). The back-tracing will, of course, take time $O(k)$, where k is the number of faces crossed by the optimal path. We therefore have the following theorem.

THEOREM 8.3. *After preprocessing, the length of the shortest path from s to any point* $x \in \mathcal{S}$ *on polyhedral surface* $\mathcal{S}$ *can be found in time* $O(\log n)$ *and a shortest path can be determined in time* $O(k + \log n)$, *where k is the number of faces crossed by the path. The preprocessing can be done in time* $O(n^2 \log n)$ (*and space* $O(n^2)$).

9. Conclusion. We have given an $O(n^2 \log n)$ algorithm for subdividing the surface of an arbitrary polyhedron so that the length of the shortest path from a given source s to any point t on the surface may be obtained simply by locating t in the subdivision. The algorithm requires $O(n^2)$ space to run. Point location can be done by standard methods in time $O(\log n)$, after which the actual path may be back-traced in time $O(k)$, where k is the number of faces traversed by the shortest path.

Our algorithm utilizes the continuous Dijkstra technique. The method is very analogous to Dijkstra's original shortest path algorithm, but includes the concepts of an expanding *wavefront* and discrete *events*. It is, in a sense, a cross between Dijkstra's algorithm and the plane sweep paradigm of computational geometry.

It is easy to extend our algorithm to the case of multiple sources and thereby to solve the Voronoi problem on the surface of a polyhedron [11]. We simply initialize the original event queue to be the events associated with the edges opposite to each source point, and then proceed exactly as before to propagate intervals. For m source points lying on the surface of a polyhedron, the algorithm builds the Voronoi diagram in time $O(N^2 \log N)$ and space $O(N^2)$, where $N = \max\{n, m\}$. One can then locate the closest source to a query point in time $O(\log N)$, and one can backtrace the path to the closest source in time proportional to the number of bends along the path.

It is also not hard to imagine how our algorithm might be generalized to other types of surfaces (instead of just those that are polyhedral). If we have a surface which is specified by giving the surface patches which make up its faces, then we can calculate for a fixed complexity of surface type (provided it is "well-behaved") how geodesic paths behave on that surface. For example, on planar patches we get the problem we have covered here, and the geodesic paths are straight lines on each face. For a patch of spherical surface, geodesic paths will be arcs of great circles for the underlying sphere. We would then have to generalize the local optimality criterion that geodesic paths that cross patch boundaries unfold into straight lines: Geodesic paths will unfold such that *locally* (at the boundary of surface patches) they unfold to be straight. This is simply a continuity condition on the slope of geodesic paths. The continuous Dijkstra algorithm allows the propagation of a wavefront across the surface patch boundaries, while maintaining the local optimality criterion. Some work is needed to figure out the details of this generalization.

We would like to be able to improve the running time of our algorithm. It may be possible to reduce the time complexity to $O(n^2)$ (the current best time for planar obstacle avoidance algorithms), but we should not hope to improve it further until the planar case is solved more efficiently.

It should be possible, however, to reduce the space requirement from $O(n^2)$. Our algorithm is wasteful in that it maintains the points of intersection of the shortest path map with each edge, and there can be a quadratic number of such points. But the actual size of the graph whose embedding on the surface yields the subdivision is only linear (in the number of edges of the original polyhedron). It should be possible to keep track of bounding hyperbolas (or lines) instead of endpoints of intervals, and then to propagate the hyperbolas across new edges without creating new structures all the time. (This also suggests that we could have allowed the algorithm to build the entire subdivision (including the partitioning of the interiors of faces) "on the fly," keeping track of the events of crossing hyperbolic boundaries, just as we did in the procedure described in § 8.) We must be careful, though, that the subdivision we build is stored in such a way that point location queries can be answered efficiently (e.g., in time $O(\log n)$). Mount [12] has shown that a subdivision can be built for a polyhedral surface and stored in a data structure of size $O(n \log n)$, and that point location queries can then be answered in time $O(\log n)$. Can this construction be built "on the fly" during the execution of continuous Dijkstra? Note that for the special case of convex polyhedra, Mount [10] has been able to reduce the space requirements to $O(n \log n)$.

Another obvious open problem is to find an efficient algorithm for determining shortest paths in three dimensions in the presence of a general collection of polyhedral obstacles. The geodesic problem is one special case in which an efficient algorithm exists, but we conjecture certain that there are other special cases which may have enough structure to yield simple efficient algorithms.

Acknowledgment. We would like to thank the referees for a careful reading and helpful suggestions for the exposition.

REFERENCES

[1] E. W. DIJKSTRA, *A note on two problems in connection with graphs*, Numer. Math., 1 (1959), pp. 269-271.
[2] M. R. GAREY, D. S. JOHNSON, F. P. PREPARATA AND R. E. TARJAN, *Triangulating a simple polygon*, Inform. Process. Lett., 7 (1978), pp. 175-179.
[3] D. G. KIRKPATRICK, *Optimal search in planar subdivisions*, this Journal, 12 (1983), pp. 28-35.
[4] D. T. LEE, *Proximity and reachability in the plane*, Ph.D. thesis, Technical Report ACT-12, Coordinated Science Laboratory, Univ. of Illinois, Chicago, IL, November 1978.
[5] D. T. LEE AND F. P. PREPARATA, *Euclidean shortest paths in the presence of rectilinear boundaries*, Networks, 14 (1984), pp. 393-410.
[6] T. LOZANO-PEREZ AND M. A. WESLEY, *An algorithm for planning collision-free paths among polyhedral obstacles*, Comm. ACM, 22 (1979), pp. 560-570.
[7] L. A. LYUSTERNIK, *Shortest Paths: Variational Problems*, Macmillan, New York, 1964.

[8] J. S. B. MITCHELL, *Shortest paths in the plane in the presence of obstacles*, Manuscript, Dept. of Operations Research, Stanford Univ., Stanford, CA, 1984.

[9] ———, *Planning shortest paths*, Ph.D. thesis, Dept. of Operations Research, Stanford Univ., Stanford, CA, August, 1986. (Available as Research Report 561, Artificial Intelligence Series, No. 1, Hughes Research Laboratories, Malibu, CA.)

[10] D. M. MOUNT, *On finding shortest paths on convex polyhedra*, Technical Report 1495, Dept. of Computer Science, Univ. of Maryland, Baltimore, MD, 1985.

[11] ———, *Voronoi diagrams on the surface of a polyhedron*, Technical Report 1496, Dept. of Computer Science, Univ. of Maryland, Baltimore, MD, 1985.

[12] ———, *Storing the subdivision of a polyhedral surface*, Proc. 2nd ACM Symposium on Computational Geometry, Yorktown Heights, NY, June 2-4, 1986.

[13] J. O'ROURKE, S. SURI AND H. BOOTH, *Shortest paths on polyhedral surfaces*, Manuscript, The Johns Hopkins Univ., Baltimore, MD, 1984.

[14] C. H. PAPADIMITRIOU, *An algorithm for shortest-path motion in three dimensions*, Inform. Process. Lett., 20 (1985), pp. 259-263.

[15] F. P. PREPARATA, *A new approach to planar point location*, this Journal, 10 (1981), pp. 473-482.

[16] J. H. REIF AND J. A. STORER, *Shortest paths in Euclidean space with polyhedral obstacles*, Technical Report CS-85-121, Computer Science Dept., Brandeis Univ., Waltham, MA, April, 1985.

[17] M. SHARIR AND A. SCHORR, *On shortest paths in polyhedral spaces*, this Journal, 15 (1986), pp. 193-215.