Discrete Comput Geom 2:153–174 (1987)

Discrete & Computational
Geometry

Storing the Subdivision of a Polyhedral Surface*

David M. Mount

Department of Computer Science and Institute for Advanced Computer Studies,
University of Maryland, College Park, MD 20742, USA

Communicated by David Dobkin

Abstract. A common structure arising in computational geometry is the subdivision of a plane defined by the faces of a straight-line planar graph. We consider a natural generalization of this structure on a polyhedral surface. The regions of the subdivision are bounded by geodesics on the surface of the polyhedron. A method is given for representing such a subdivision that is efficient both with respect to space and the time required to answer a number of different queries involving the subdivision. For example, given a point x on the surface of the polyhedron, the region of the subdivision containing x can be determined in logarithmic time. If n denotes the number of edges in the polyhedron, m denotes the number of geodesics in the subdivision, and K denotes the number of intersections between edges and geodesics, then the space required by the data structure is $O((n+m)\log(n+m))$, and the structure can be built in $O(K+(n+m)\log(n+m))$ time. Combined with existing algorithms for computing Voronoi diagrams on the surface of polyhedra, this structure provides an efficient solution to the nearest-neighbor query problem on polyhedral surfaces.

1. Introduction

In computational geometry there has been considerable study of algorithms for problems involving points and lines in two-dimensional and three-dimensional Euclidean space. In applications such as robotics, computer-aided design, and terrain modeling it is frequently more meaningful to consider problems on

* A preliminary version of this paper appeared at the Second Annual Symposium on Computational Geometry. The support of the Air Force Office of Scientific Research under Contract F-49620-85-K-0009 is gratefully acknowledged.

surfaces, that is, subsets of three-dimensional space that locally are continuous deformations of the plane. These surfaces are frequently approximated piecewise by polyhedra. For example, in [16] it was shown how to compute the shortest path between two points on the surface of a convex polyhedron in $O(n^3 \log n)$ time. This result was improved to finding both shortest paths and Voronoi diagrams on the surface of a polyhedron in $O(n^2 \log n)$ time [9], [11]. These shortest path problems (when paths are constrained to lie on the surface of a polyhedron) are of interest in areas such as autonomous vehicle navigation, when hilly terrain is being modeled.

A number of problems in computational geometry that have been solved on the plane can be meaningfully posed on a polyhedral surface. We consider the polyhedral generalization of the point location problem. The Point Location Problem on the plane is: given a polygonal subdivision Q of the plane into regions bounded by straight line segments, and given a query point x, find the region of Q that contains x. By a ***subdivision*** we man a finite set of regions covering the plane whose interior are pairwise disjoint. A well-known application of the point location problem is the location of a query point among the regions of a Voronoi diagram to answer the nearest-neighbor query [14]. In light of the results in [9] and [11] on computing Voronoi diagrams on polyhedral surfaces, it is natural to consider the point location problem on the surface of a polyhedron.

The planar version of the point location problem has been well studied, and solutions are known that are asymptotically optimal with respect to preprocessing time and the amount of storage needed to answer queries [4], [7]. This problem can be posed on polyhedral surfaces, where the straight line edges in the plane are replaced by geodesics. A curve on a polyhedron is *geodesic* if it is locally a shortest path. We consider the following problem:

Polyhedral Surface Point Location. Given a subdivision Q partitioning the surface of a polyhedron into regions bounded by geodesics, and given a query point x on the surface of the polyhedron, find the region of Q that contains x.

We assume that the subdivision Q is defined by a graph embedded on the surface of the polyhedron so that the geodesic edges of Q do not intersect each other except possibly at their endpoints. The query point x on the surface of the polyhedron is represented by giving the face of the polyhedron containing x and the coordinates of x with respect to a coordinate frame attached to the plane containing the face.

A geodesic on a polyhedral surface is the natural generalization of a straight line segment on the plane. A geodesic traverses a face of the polyhedron in a straight line. A geodesic crosses an edge of the polyhedron so that if the faces about the edge are "unfolded" to lie in a common plane, then the geodesic becomes a straight line segment in the vicinity of the edge. Geodesics cross vertices in a manner that is not as locally predictable (see [9] and [11]), hence we will assume that geodesics of the subdivision intersect vertices only at their endpoints. Just as line segments and hyperbolic segments arise in Voronoi diagrams of points and circles in the plane [15], geodesics and their hyperbolic generalizations arise in computing Voronoi diagrams on convex and nonconvex polyhedra, respectively. Since geodesics may generally be quite complex, we make the simplifying assumption that no geodesic traverses a given face more than once. This assumption is satisfied by the geodesics arising in Voronoi diagrams on the surface of polyhedra, and in general it can always be met by the insertion of additional subdivision vertices.

There is an obvious solution to the polyhedral point location problem; namely, solve the point location problem separately for each face of the polyhedron by existing planar point location methods. The amount of space required to store the resulting data structures will be proportional to the number of intersections between geodesics and polyhedron faces. Thus, if n is the number of edges on the polyhedron and m is the number of geodesics, the space requirements could be as large as $O(nm)$. This is far from optimal, since each geodesic is determined by a constant amount of information: an endpoint, its initial direction, and its length. This follows from the locally predictable manner in which geodesics traverse faces and edges of the polyhedron.

Our main result is the development of a space-efficient representation of a geodesic subdivision on the surface of a polyhedron. Given this representation, a point location query can be answered in $O(\log(n+m))$ time. The amount of storage required is $O((n+m)\log(n+m))$, that is, a logarithmic factor from optimal. Let K denote the number of intersections between geodesics and polyhedron edges (K is $O(nm)$). The representation can be built in time $O(K+(n+m)\log(n+m))$. This representation is of value even if the point location problem is not the object of study, since many natural queries can be answered quickly while avoiding the brute-force $O(nm)$ storage cost. An example is the problem of listing the geodesics traversing a given edge. Using our structure, this problem can be answered in optimal time, that is, time linear in the size of the output.

It seems at first that standard plane sweep methods [2] used for representing polyhedra, should suffice to store the geodesic/polyhedron edge intersection information efficiently. However, the unpredictable nature of geodesics makes it quite difficult, in general, to sweep in a fixed direction without encountering the same geodesic repeatedly. That is, unlike line segments, geodesics fail to act monotonically. Our method employs a hierarchical decomposition of the set of geodesic and polyhedron edge intersections. This method is similar in flavor to Kirkpatrick's hierarchical triangular decompositon of a planar subdivision [7], but the specifics of our algorithm are quite different.

The data structure presented here to represent the structure of the polyhedron subdivision is a collection of trees, one tree for each edge in the polyhedron. The leaves of the tree associated with a given polyhedron edge e correspond to the geodesics that traverse e, ordered from left to right. Each subtree has $O(\log(n+m))$ height. By sharing common subtrees the total space requirements are reduced. The problem of point location is reduced to the problem of a searching in a tree of logarithmic depth. The problem of listing the geodesics that traverse a given edge reduces to a simple tree traversal.

The paper is organized as follows. Section 2 contains definitions and notation. Section 3 describes the procedure by which the discrete combinatorial part of the data structure is constructed. In Section 4 the addition of geometric information to the data structure is presented, and query processing is discussed.

2. Definitions and Notation

The notion of a *sequence* will be central to our presentation. Define an *ordered sequence* $(x_1, x_2, \ldots, x_j)$ to be a finite list of linearly ordered elements. Define a *bi-ordered sequence* $\langle x_1, x_2, \ldots, x_j \rangle$ to be a finite list of linearly ordered elements in which no distinction is made between the sequence and its reverse. The *endpoints* of the sequence are x_1 and x_j. Unless otherwise noted, the sequences that we consider will be bi-ordered sequences. For sequences S_1 and S_2, we say that S_1 is a *subsequence* of S_2 if S_1 occurs as a contiguous sequence within S_2. Let $S_1 = \langle x_1, x_2, \ldots, x_j \rangle$ and let $S_2 = \langle y_1, y_2, \ldots, y_k \rangle$. Assuming that the elements of S_1 and S_2 are disjoint, their *concatenation*, $S_1 + S_2$, is defined by specifying the endpoints at which the sequences are joined. For example, the *concatenation* of S_1 and S_2 at x_1, y_1 is the sequence $\langle x_j, \ldots, x_1, y_1, \ldots, y_k \rangle$.

Let P denote a polyhedron, which consists of a finite set of vertices, edges, and convex polygonal faces. We are not concerned with P's realization as a volume in 3-space, but rather we only consider the graph-theoretic specification of P as a planar graph embedded on a surface of genus 0; see [6] and [12] for example. Let n denote the number of edges in P. By Euler's formula and the planarity of P, n is bounded by a linear function of the number of vertices in P. (In general, our results can be extended to graphs embedded on an arbitrary orientable 2-manifold. Our complexity bounds hold if the genus of the manifold is bounded by a linear function of the size of the graph's vertex set. This requirement is met by the polyhedra used in CAD applications.)

For the purpose of extracting discrete incidence information, such as listing the geodesics that traverse an edge, no geometric information is needed. However, if we wish to solve the point location problem, then additional geometric information is added to specify the location of a point on a face. Each face is associated with a two-dimensional coordinate system. A point located on face f of the polyhedron is represented by specifying f and giving the point's coordinates with respect to f's coordinate system.

Let Q denote a geodesic subdivision of P. Q is represented by a graph embedded on the surface of P whose edges (the geodesics) do not intersect one another except possibly at their endpoints (the vertices of Q). Henceforth, we use the term geodesic to refer to the edges of Q to avoid confusion with the edges of the polyhedron. Let m denote the number of geodesics in Q. Since P is planar, Q is also planar. If Q has no multiple edges then, by Euler's formula, m is bounded by a linear function of the number of vertices in Q. Each vertex of Q is represented as any point on P's surface, by giving a face and its coordinates with respect to the face.

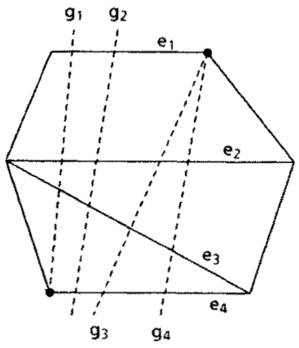

Fig. 1. Intersection sequences.

For a geodesic g, let $I(g)$ denote the bi-ordered sequence of edges intersecting g given in order from one end of g to the other. Similarly, for an edge e, let $J(e)$ denote the bi-ordered sequence of geodesics intersecting e given in order from one end of e to the other.

We make the following assumptions to handle degenerate cases. If a vertex of Q lies on an edge of P, we add a vertex to P at this point and split the edge into two edges. If a vertex of Q coincides with a vertex of P, then the geodesics of Q incident on this vertex are considered to intersect the nearest edge e of P in clockwise order about the vertex. The order of intersection of these geodesics in $J(e)$ is in clockwise angular order about the vertex, so that the furthest counterclockwise geodesic is an endpoint of $J(e)$. $I(g)$ and $J(e)$ can be constructed simultaneously using time and space proportional to the number of edge/geodesic intersections by a simple traversal of Q. For example, in Fig. 1 polyhedron edges are denoted by solid lines and geodesics with dashed lines. The edges have been unfolded so the faces lie on a common plane, and geodesics are straight line segments. $J(e_1)=\langle g_1, g_2, g_3, g_4\rangle$ and $I(g_1)=\langle e_1, e_2, e_3, e_4\rangle$.

We say that two geodesics g_1 and g_2 are *adjacent* on an edge e if g_1 and g_2 appear consecutively in $J(e)$. Similarly, two edges e_1 and e_2 are *adjacent* on a geodesic g if they appear consecutively in $I(g)$. We cross reference the entries $e \in I(g)$ with $g \in J(e)$ to facilitate local traversals of the structure. Each geodesic is represented by giving its endpoints (vertices of Q) and $I(g)$. By unfolding the edges of $I(g)$, it is easy to trace the path of the geodesic.

3. The Incidence Structure

As mentioned in the introduction, the relationship between the polyhedron P and the subdivision Q is represented by a data structure consisting of a set of trees.

Definition. An *incidence structure* for Q on P is a collection of ordered binary trees with shared subtrees whose roots correspond 1-1 with the edges of P, and whose leaves correspond 1-1 with the geodesics of Q, such that for each edge e of P, the leaves of the corresponding subtree rooted at e form the sequence $J(e)$.

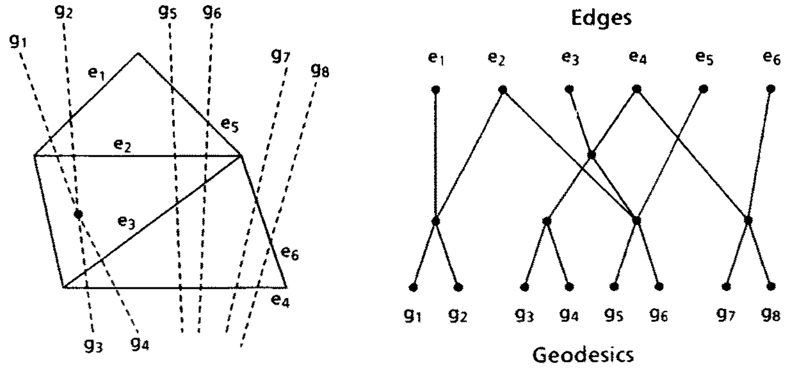

Fig. 2. An incidence structure.

The precise sense in which these trees are ordered is described in the next section. For example, in Fig. 2 we show a possible incidence structure for a set of edges and geodesics. The descendants of edge e_3 are $\langle g_3, g_4, g_5, g_6\rangle = J(e_3)$. The main result of this section is the following theorem.

Theorem 3.1. *Consider a polyhedron P with n edges and a subdivision Q bounded by m geodesics embedded on P so that no two geodesics cross each other and no geodesic crosses an edge of P more than once. An incidence structure of height $O(\log(n+m))$ containing $O((n+m)\log(n+m))$ nodes can be built in $O(K+(n+m)\log(n+m))$ time, where K denotes the number of intersections between geodesics and edges.*

Each node in the structure can be implicitly associated with the set of geodesics that are its descendants at the leaf level and with the set of edges that are its ancestors at the root level. To help describe this association, we introduce the concept of a *bundle* of edges and geodesics.

Definition. *A bundle* is a pair (G, E) where G is a sequence of geodesics, and E is a sequence of edges, such that:

1. G is a subsequence of $J(e)$ for all $e \in E$,
2. E is a subsequence of $I(g)$ for all $g \in G$.

A bundle (G, E) can also be thought of as a subset of intersection points between geodesics and polyhedron edges, namely the set of intersections (g, e) in the product $G \times E$. The definition of a bundle implies that each such geodesic/edge pair intersect. In Fig. 3 the geodesic/edge intersections of the shaded region form a bundle. Intuitively a bundle corresponds to a four-sided region of the polyhedron bounded oppositely by two edges and by two geodesics within which there are neither any vertices from P nor vertices from Q. Define the *g-length* of a bundle to be the number of geodesics in G, and define the *e-length* to be the number of edges in E. The *g-endpoints* and *e-endpoints* of a

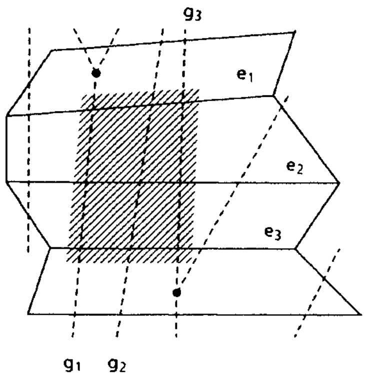

Fig. 3. The bundle $(\langle g_1, g_2, g_3\rangle, \langle e_1, e_2, e_3\rangle)$.

bundle are the endpoints of the corresponding sequences. A bundle is *g-maximal* if G cannot be extended to a larger sequence (*e-maximal* is defined similarly). There are three special types of bundles that are significant to our presentation.

Definition.

1. For each edge e, $(J(e), \langle e\rangle)$ is the *single edge* bundle for e.
2. For each geodesic g, $(\langle g\rangle, I(g))$ is the *single geodesic* bundle for g.
3. For each pair of geodesics g_1 and g_2 that are adjacent along some edge, let E be a maximal common subsequence of edges along which g_1 and g_2 are adjacent. The *double geodesic* bundle for g_1 and g_2 along E is $(\langle g_1, g_2\rangle, E)$. We denote the set of all double geodesic bundles by D.

A collection of bundles is called an *incidence partition* if the associated subsets of geodesic/edge intersections form a partition of all the geodesic/edge intersections. The collection of single geodesic bundles and the collection of single edge bundles are both examples of incidence partitions. The set of double geodesic bundles does not form an incidence partition because a given geodesic/edge intersection may belong to two different double geodesic bundles.

The construction of the incidence structure is linked to certain operations performed on bundles. Each node in the structure is associated with a bundle. For example, the leaf node corresponding to a geodesic g is, in fact, associated with the single geodesic bundle for g, and the root node corresponding to an edge e is associated with the single edge bundle for e. We build the incidence structure in a bottom-up fashion, beginning with the leaves (the geodesics) and working up to the roots (the edges). The fundamental operation performed on bundles is a merging operation that we call *tying*. In this operation, two bundles are merged and replaced by one or more new bundles of greater g-length but possibly smaller e-length. Beginning with the single geodesic bundles we tie bundles together in pairs, gradually transforming the incidence partition from a collection of single geodesic bundles into a collection of single edge bundles. The incidence structure is essentially an operator tree, recording the relationships

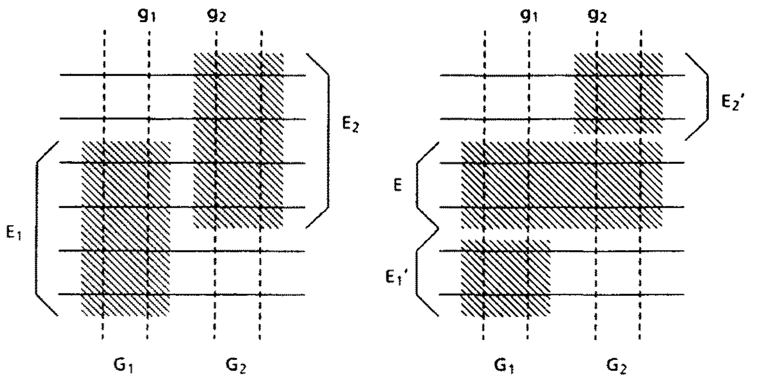

Fig. 4. Tying bundles.

among bundles as tying occurs. Thus, the incidence structure is built in a bottom-up manner, with one new node added whenever a new bundle is formed.

To describe the tying operation more formally, let $b_1=(G_1, E_1)$ and $b_2=(G_2, E_2)$ be two bundles for which there are two g-endpoints $g_1 \in G_1$ and $g_2 \in G_2$ that are adjacent on some edge e that is in both E_1 and E_2; see Fig. 4. Intuitively, the *tie* of b_1 and b_2 at e consists of concatenating b_1 and b_2 along a maximal common subsequence of edges into a new bundle. Both b_1 and b_2 are then replaced by this new bundle together with any of their leftover fragments.

Algorithm 3.1. Tying bundles b_1 and b_2 at edge e.

1. Let E be a maximal subsequence of E_1 and E_2 containing e on which g_1 and g_2 are adjacent. Partition E_1 into at most three subsequences, E_1', E, and E_1''. (The first and third subsequences may be empty.) Similarly, partition E_2 into at most three subsequences, E_2', E, and E_2''.
2. Create new bundles (G_1, E_1') and (G_1, E_1''). Do the same for b_2.
3. Create the new bundle (G_1+G_2, E), where the geodesic sequences are concatenated at g_1 and g_2.
4. Delete the bundles b_1 and b_2.

Note that if a collection of bundles forms an incidence partition before a tying operation is performed, then the new collection of bundles also forms an incidence partition. After tying, the number of bundles in a partition increases by at most 3. Because it consists of simple list operations, tying can be performed in time proportional to the length of E.

The tying operation is recorded as follows in the incidence structure. Let us assume that each node in the incidence structure is named by its corresponding bundle. For each newly constructed bundle, we add a new node to the incidence structure. The nodes (G_1, E_1') and (G_1, E_1'') have the node b_1 as their only child (and similarly for b_2), and the node (G_1+G_2, E) has both b_1 and b_2 as children. In the next section we discuss the issue of which node is the left child and which is the right child. Note that if the bundle (G', E') is the parent of some bundle

(G, E), then G is a subsequence of G' and E' is a subsequence of E. For example, referring back to Fig. 2, the single geodesic bundles for g_7 and g_8 (the rightmost leaves) are tied along $\langle e_4, e_6\rangle$ forming their parent bundle $(\langle g_7, g_8\rangle, \langle e_4, e_6\rangle)$. This bundle is then split when it is tied to the bundle $(\langle g_3, g_4, g_5, g_6\rangle, \langle e_3, e_4\rangle)$ (the central node) along edge e_4 forming the three single edge bundles (roots) $(\langle g_3, \ldots, g_6\rangle, \langle e_3\rangle)$, $(\langle g_3, \ldots, g_8\rangle, \langle e_4\rangle)$, and $(\langle g_7, g_8\rangle, \langle e_6\rangle)$. The remainder of the incidence structure shown in Fig. 2 can be built using this operation along with the simple enhancement of removing tree nodes that have but one child.

As mentioned earlier, the incidence structure is built bottom-up starting with the single geodesic bundles and then repeatedly tying until only single edge bundles remain. The order in which we choose to tie the bundles is critical to the efficiency of the structure. There are two constraints that must be met to ensure this efficiency. First, we wish to keep the number of bundles in the incidence partition as small as possible, since this directly influences space requirements. Haphazardly tying bundles may cause significant fragmentation in the incidence partition, which can result in as many as $K = O(nm)$ bundles. Second, we must tie bundles in a relatively balanced manner to guarantee that the height of the incidence structure will be logarithmic. The issues of avoiding *fragmentation* and guaranteeing *balance* will be the subject of the rest of this section.

In order to avoid fragmentation, we introduce a modification of the tying operation, called a *full tie*. A single full tie operation will actually consist of a series of individual tying operations. To define this operation, first recall the set D of double geodesic bundles. The set D will serve as an aid in defining the tying process, and is independent of the bundles represented by the nodes in the incidence structure. To avoid confusion we use the term "bundle" for bundles represented by nodes in the incidence structure, and we use "an element of D" for these special bundles.

Suppose that we have built a set of bundles that forms an incidence partition. Let $d = (\langle g_1, g_2\rangle, E)$ be an element of D, such that for each $e \in E$, (g_1, e) and (g_2, e) lie in different bundles of the partition. We define the *full tie* along d to be the tie of all bundles that meet along the length of d. The complete description is given in Algorithm 3.2, and Fig. 5 illustrates the operation. As with regular ties, a full tie is an operation that preserves incidence partitions.

Algorithm 3.2. The full tie along $(\langle g_1, g_2\rangle, E)$.

1. Let $b_1, b_2, \ldots, b_j$ $(j \geq 1)$ be the bundles that contain the intersections $\{g_1\} \times E$, and let $c_1, c_2, \ldots, c_k$ $(k \geq 1)$ be the bundles that contain the intersections $\{g_2\} \times E$. The b_i and c_h are ordered according to their incidence with E.
2. Both b_1 and b_j may extend beyond E. If so, split off this extension. This may result in the creation of at most two new bundles, and b_1 and b_j are trimmed back. Now, the edge sequence of every bundle b_i, $1 \leq i \leq j$, is a subsequence of E. We do the same with c_1 and c_k.
3. The bundles b_i define a partition of E into j subsequences, and the bundles c_i partition E into k subsequences. For each b_h and c_i that are adjacent on

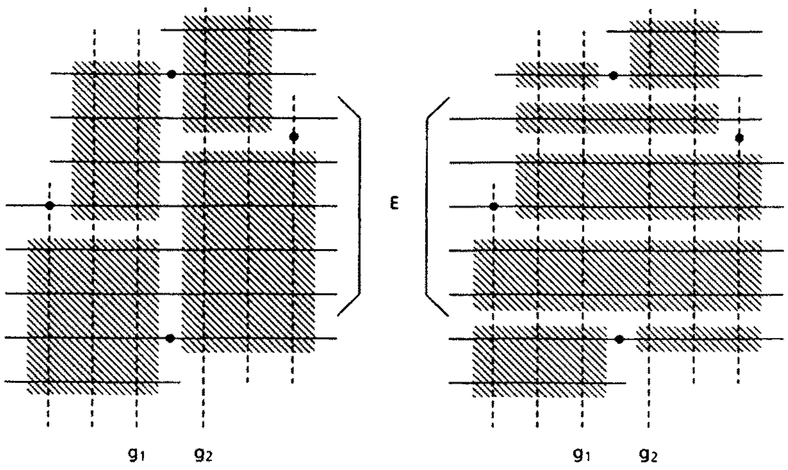

Fig. 5. The full tie along $(\langle g_1, g_2\rangle, E)$.

some edge $e \in E$, we form a new bundle by tying b_h and c_i at edge e. This results in the replacement of the original $j+k$ bundles by at most $j+k-1$ new bundles.

The effect of this operation on the incidence structure is the same as if each pair of bundles b_h and c_i were tied individually. The full tie along d replaces $j+k$ existing bundles with at most $j+k+3$ new bundles ($j+k-1$ along E and at most four leftover extensions). The operation can be performed in time proportional to the e-length of d. Note that a full tie can be performed at most once for each $d \in D$, since, after tying, all of the geodesic/edge intersections in d adjacent along each edge appear in the same bundle. The full tie operation along $d = (\langle g_1, g_2\rangle, E)$ assumes that the geodesic/edge intersections $(\langle g_1\rangle, E)$ appear in a different bundle from their counterparts $(\langle g_2\rangle, E)$. If we start with the single geodesic bundles and perform full ties along the elements of D, this assumption will always be met because a pair of geodesic/edge intersections that are adjacent on an edge can be placed into the same bundle by the one and only one element of D that contains this pair. Thus, we can perform full ties along every element of D. After performing full ties along all of the elements of D, there are no more bundles that can be tied, so each edge will be traversed by at most bundle.

By performing full ties along the elements of D it can be proved (Lemmas 3.4 and 3.3 below) that fragmentation of bundles caused by haphazard tying will be avoided. To guarantee that bundles are tied in a balanced way to ensure logarithmic height in the incidence structure, we structure the order in which full ties are performed. We organize the tying process in stages. At each stage, we select a subset of elements from D, say S, and perform full ties along all of the elements of S. The elements of S are removed from consideration for future tying. At stage k, we let D_k denote the elements of D that remain eligible to use in tying.

Balance will be achieved by maintaining the restriction that, within a single stage, two bundles are tied if and only if they were both formed at earlier stages. This restriction limits the subsets $S \subseteq D_k$ that can be selected for tying at stage k. We say that a subset of D_k is *eligible* for tying if it satisfies this condition. The problem reduces to determining a sufficiently large eligible subset of D_k. Consider a bundle $b = (G, E)$ in the incidence partition formed prior to stage k. Let g_1 and g_2 be the g-endpoints of G and let $e \in E$. If the geodesic/edge intersection (g_1, e) is involved in tying during stage k, then the intersection (g_2, e) is ineligible for tying in stage k (and vice versa). This, if d_1 and d_2 are elements of D_k, such that $(g_1, e) \in d_1$ and $(g_2, e) \in d_2$, then d_1 and d_2 cannot both be eligible to be used in tying at stage k. This suggests the following relation defined on the elements of D_k.

Definition. Consider the bundles defining an incidence partition that result after performing some series of full ties. Let D_k denote the elements of D that remain to be tied. Two elements d_1 and d_2 in D_k are said to *conflict* if there exist geodesic/edge intersections $(g_1, e) \in d_1$ and $(g_2, e) \in d_2$ that lie in the same bundle. The *conflict graph* for D_k is an undirected graph with vertex set D_k whose edges are the conflicting pairs of D_k (see Fig. 6).

Note that the geodesics g_1 and g_2 in the definition of the conflict graph are the g-endpoints of their common bundle because they are both contained in elements of D_k which, by definition, have not yet been tied. It follows from the previous discussion that a subset S of D_k is eligible to be used in tying if and only if it is a subset of vertices in the conflict graph that are not pairwise adjacent, that is, an *independent set* in the conflict graph. The algorithm for tying bundles follows immediately.

Algorithm 3.3. Tying bundles by independent sets.

```
1. Initially the set of bundles consists of the single geodesic bundles.
2. Let $D_1 = D$, be the set of double geodesic bundles. Let $k = 1$.
3. While $D_k$ is nonempty repeat steps 4-7.
4. Form the conflict graph for $D_k$ from the existing incidence partition.
```

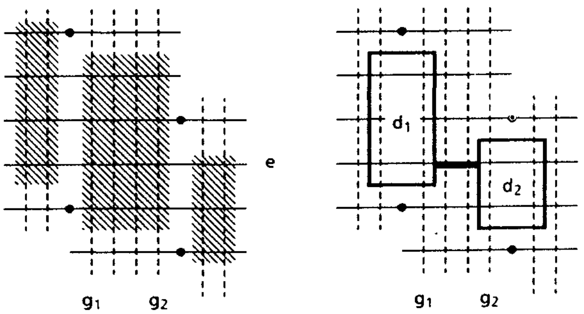

Fig. 6. The conflict graph.

5. Compute an independent set S in the conflict graph for D_k.
6. Perform full ties along each $d \in S$ and construct the corresponding nodes in the incidence structure.
7. Let $D_{k+1} = D_k - S$. Let $k = k+1$.
8. At this point, every edge is traversed by at most one bundle. For each bundle $b = (G, E)$ where $|E| > 1$, and for each $e \in E$, create the single edge bundle $b_e = (G, \langle e \rangle)$, and make b the only child of b_e in the incidence structure.

Each iteration of the whole loop in Algorithm 3.3 constitutes one stage. The next fact is an important observation in finding an independent set in the conflict graph.

Lemma 3.1. *At each state of Algorithm* 3.3, *the conflict graph is a planar graph.*

Proof. We show how to draw the conflict graph on the polyhedron P so that its edges do not cross. Let k denote the current stage. For each $d = (\langle g_1, g_2 \rangle, E) \in D_k$, identify d with a four-sided region on the surface of the polyhedron bounded by g_1, g_2 and by the endpoint edges of E. These regions, called *tie regions*, have pairwise disjoint interiors because geodesics and edges do not cross their own kinds, and because of the e-maximality of elements of D.

The tie regions are in 1-1 correspondence with the vertices of the conflict graph at stage k. The edges between conflict graph vertices, or *conflict edges* can be drawn as follows. If there is an edge between d_1 and d_2 in the conflict graph for D_k, then by definition there is a bundle b in the incidence partition with g-endpoints g_1 and g_2 and an edge e in b such that $(g_1, e) \in d_1$ and $(g_2, e) \in d_2$. We draw the conflict edge from (g_1, e) to (g_2, e) along the edge e. No other conflict edge can overlap this portion of edge e, because the bundles of the incidence partition are disjoint. Furthermore, no conflict edge can cross this portion of e, because all conflict edges lie on polyhedron edges, which do not cross each other. Finally, this conflict edge cannot cross a tie region, because the tie regions span the regions between untied bundles and the conflict edge lies entirely within the region defined by the bundle. Since $(g_1, e) \in d_1$ and $(g_2, e) \in d_2$, the edge endpoints touch the boundary of the tie regions for d_1 and d_2. The planar graph results by continuously contracting each tie region boundary into a single point, pulling along the conflict edges on its boundary. $\square$

It is easy to prove that planar graphs have easily computable independent sets whose size is at least a constant fraction of the size of their vertex set (see [7]). This follows, for example, by the fact that a 5-coloring of a planar graph can be computed in time linear in the number of vertices [1]. The largest color class is an independent set of size at least $v/5$. The fact that there exists a constant $c \geq 1$, such that an independent set can be found in the conflict graph of size at least v/c, implies that $|D_k| \leq |D_{k-1}|(c-1)/c$. Thus, after a logarithmic number of stages $|D_k| = 0$ and the process terminates. Since at each stage at most one new level is added to the structure, we have the following result.

Lemma 3.2. *The incidence structure built by Algorithm* 3.3 *has* $O(\log|D|)$ *height.*

Lemma 3.3. *The number of elements in* D *is* $O(n+m)$.

Proof. To compute the size of D we place an upper bound on the number of e-endpoints of the elements of D. Let g_1 and g_2 be two geodesics that are adjacent on an edge e. There is exactly one element of D that contains both of the intersections of g_1 and g_2 along e. This element of D can be defined by expanding e to a maximal subsequence of edges along which g_1 and g_2 are adjacent. There are two reasons why this expansion may terminate. First, we reach an endpoint of either g_1 or g_2. The number of such terminations is at most $4m$ because each geodesic has two endpoints, and each geodesic may be adjacent to at most two other geodesics (one on the right and one on the left) along the last edge traversed by the geodesic; see the top of Fig. 7(a).

The second reason for termination is that we reach an edge e' where g_1 and g_2 are adjacent, but the neighboring edges of e' in $I(g_1)$ and $I(g_2)$, say e_1 and e_2, are not equal. The edges e', e_1, and e_2 lie on a common face f of the polyhedron; see Fig. 7(b). We have a configuration in which two geodesics enter a face adjacently through a common edge and diverge, leaving the fact through two different edges. Charge this termination to a vertex v causing the divergence, that is, a vertex lying between e_1 and e_2. Because geodesics do not cross one another, no other pair of geodesics will be charged to this vertex within the face f. This implies that the number of charges made to any vertex is no greater than the number of faces incident on that vertex. Thus, the total number of terminations of this second type is bounded by the sum of degrees of the vertices of the polyhedron, which is twice the number of edges or $2n$. Therefore, the total number of e-endpoints is at most $4m+2n$ which is $O(n+m)$. □

Lemma 3.4. *Let* $N=n+m$. *The number of nodes in the incidence structure built by Algorithm* 3.3 *is* $O(N\log N)$.

Proof. Initially there are m nodes in the incidence structure corresponding to the m geodesics, and at the root level there are n nodes corresponding to the

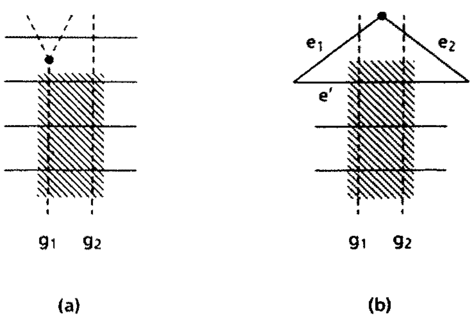

Fig. 7. Termination of an element in D.

edges of the polyhedron. Since there are logarithmically many stages (in $|D| = O(N)$) to show that the remaining number of nodes is $O(N \log N)$, it suffices to show that each stage of the algorithm creates at most N new nodes. Recall from the remarks following Algorithm 3.2 that, whenever a full tie is performed, some number h of nodes (i.e., bundles) that were eligible for tying are replaced by at most $h+3$ new nodes. The total number of full ties that are performed is $|D|$. Let s_k denote the number of full ties performed at stage k. Thus $s_1+s_2+\cdots = |D|$. The number of nodes eligible for tying at stage k is bounded by the number of nodes eligible for tying at stage $k-1$ plus $3s_{k-1}$. Thus, the number of nodes eligible for tying and hence the number of nodes created at stage k is at most $m+3(s_1+s_2+\cdots+s_{k-1}) \le m+3|D| = O(N)$. □

The following lemma completes the proof of Theorem 3.1.

Lemma 3.5. *The incidence structure can be built in $O(K+(n+m)\log(n+m))$ time, where K denotes the number of geodesic/edge intersections.*

Proof. The running time of Algorithm 3.3 is determined by the sum of the times needed to perform the following tasks:

1. Compute the single geodesic bundles.
2. Compute D.
3. Create the nodes of the incidence structure.
4. Build the conflict graph for D_k for each stage.
5. Construct an independent set in each conflict graph.
6. Perform the full ties.

Task 1 is computed by a simple traversal of $I(g)$ for each geodesic, requiring $O(K)$ time. For Task 2 we compute the elements of D by the same expansion process described in the proof of Lemma 3.3. Each intersection (g, e) is visited at most twice in the process. Thus the time to construct D is proportional to the number of geodesic/edge intersections, which is $O(K)$. For Task 3, the number of nodes in the incidence structure is $O((n+m)\log(n+m))$ and constant time suffices to create each node. For Task 4, note that the conflict graph for D_1 can be built in $O(K)$ time, and the graph can be updated as the full tie is being performed as follows. For each geodesic/edge intersection we can determine its immediate neighbors on the edge in constant time. For each such pair we store a pointer to the element of D containinng this pair. As a full tie is performed along $d \in D$, we delete node d from the conflict graph and check the e-endpoints of the newly merged bundles for new edges to add to the graph. Thus, the time for Task 4 is the same as the time for Task 6, which we give below. For Task 5, an independent set in D_k can be constructed in linear time in $|D_k| = O(n+m)$. Since there are logarithmically many stages the total time is $O((n+m)\log(n+m))$. To determine the time to perform Task 6, recall that the time to perform a full tie along $d \in D$ is proportional to the e-length of d. The sum of e-lengths in D is $O(K)$ because each of the K geodesic/edge intersections lies in at most two members of D, and the e-length of a bundle in D is one-half its number of intersections in the bundle. □

Before ending this section, we describe one further performance enhancement that is made to the incidence structure without altering the asymptotic running time of the algorithm. Ignoring root nodes, there may be trivial nodes in the structure that have only one child. This happens, for example, when a part of a bundle is tied and the remainder is broken off to form an untied bundle. The remaining subbundle has the original bundle as its only child. We eliminate links to trivial nodes in the incidence structure by a postprocessing step that moves each link down to the first descendant node that is either a leaf or has two children. Now, except for root nodes, the nodes of the incidence structure each have either zero or two children. This implies that the size of the subtree rooted at any given node is at most twice the number of its leaves. We will use this fact in the next section on query processing.

We remark in passing that throughout this section we have made essentially no use of the fact that the polyhedron and the geodesic subdivision are geometric objects, or even that the geodesics form a subdivision. The only data that we have used in defining the incidence structure are the intersection sequences, $I(g)$ and $J(e)$, the assumption that the curves represented by these structures can be drawn on the plane so that they do not cross over curves of their same kind, and the assumption that no geodesic traverses a face more than once. Hence, the incidence structure may be valuable in other applications that involve pairs of planar objects that overlap each other [5].

4. Query Processing

In this section we consider how to answer a number of queries about the subdivision by using the incidence structure. We begin by considering the following two discrete queries.

Geodesic Incidence with an Edge. Given an edge e of the polyhedron, list the sequence of geodesics that traverse e ordered from one end of e to the other, that is, list $J(e)$.

Geodesic Incidence with a Face. Given a face f of the polyhedron, list the set of geodesics that traverse f.

Before describing the solutions to these queries, we describe some enhancements to the incidence structure that will allow us to determine the relative order in which geodesics traverse an edge. Normally, when dealing with a binary tree, there is an explicit notion of left and right subtrees. In the case of the incidence structure, this distinction is not made so easily. Consider, for example, the six

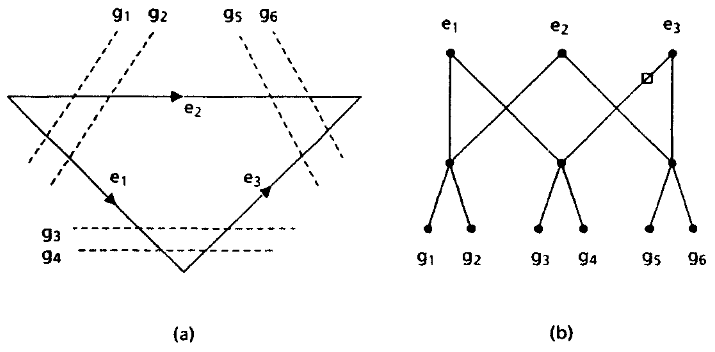

Fig. 8. Ordering subtrees of the structure.

geodesics, $g_1, \ldots, g_6$, traversing the triangular face consisting of edges e_1, e_2, e_3 shown in Fig. 8(a). By specifying that g_1 is to the left of g_2, it follows that g_3 is to the left of g_4 on edge e_1, and g_5 is to the left of g_6 on edge e_2. However, now there is no way to define right and left with respect to the edge e_3 that is simultaneously locally consistent with the other two edges.

We solve this problem by adding one additional bit to each link of the incidence structure that allows us to define left and right separately for each bundle. Each polyhedron edge is arbitrarily oriented by distinguishing the edge's endpoints as head and tail. Points along the edge are ordered from left to right as we move from the tail to the head of the edge. With each bundle $b = (G, E)$, we arbitrarily select one edge $e \in E$ called the *representative edge* for the bundle. Since all the geodesics of G traverse the edge e, this implicitly defines an order of the geodesics of G.

Let $b_1 = (G_1, E_1)$ be a bundle in the incidence structure with a parent node $b_2 = (G_2, E_2)$, and let $e_1 \in E_1$ be the representative edge for bundle b_1. This implies that E_2 is a subsequence of E_1, and G_1 is a subsequence of G_2. We arbitrarily select an edge $e_2 \in E_2$ to be the representative edge for b_2. The geodesics of G_1, considered in order from left to right along e_1, either traverse e_2 in left to right order or in right to left order. In the latter case, we place a *mark* on the link between b_2 and b_1 in the incidence structure. The mark indicates that all notion of left to right within the subtree rooted at the note b_1 is to be interpreted in reverse (as right to left). Within the subtree of b_1 there may be other marks that reverse the order again. Thus, if p is a path from a root of the incidence structure to some node, then the interpretation of left and right subtrees at this node is a function of the parity of the number of marked links in p. For example, in Fig. 8(a), the directions on the edges indicate the implied left to right order, and in (b) a mark is shown on one of the links from e_3. Thus, the order of geodesics along edge e_3 is taken to be $\langle g_4, g_3, g_5, g_6 \rangle$.

When two bundles, b_1 and b_2, are made children of a bundle b in the incidence structure, they are merged according to their relative order along e, the representative edge for b. The disjointness of bundles implies that one bundle lies entirely to the left of the other bundle with respect to e. The leftmost bundle becomes the left son of b and the other becomes the right son. If the direction of the representative edge of a child bundle, say b_1, differs from the direction of the representative edge of the parent b, then a mark is added on the link between b and b_1. This determination can be made in constant time for each tie.

The Edge Incidence Problem is now easily solved by simply performing a left to right depth first traversal of the tree rooted at edge e in the incidence structure, listing geodesics (leaves of the structure) as they are encountered. Note that the bundle associated with the root e is a single edge bundle, hence its representative edge is e. This implies that the geodesics are listed in left to right order with respect to the orientation of e. The complexity of the operation is equal to the size of the tree rooted at e. In the previous section the size of each tree was shown to be at most twice the number of its leaves. Therefore the running time is linear in the length of the output.

To solve the Face Incidence Problem, note that the geodesics incident on a given face f either intersect an edge of f or are entirely contained within f. The latter set of geodesics will not be represented at all in the incidence structure, so for each f we simply record the set of geodesics entirely contained within f. The total space required to store this information is $O(n+m)$. The remaining geodesics are found by solving the Edge Incidence Problem for each edge of f that is traversed by at least one geodesic. This set of edges that are incident on f and are traversed by at least one geodesic can be stored explicitly in $O(n)$ space. The time required to solve the problem is linear in the size of the output.

Next, we consider how to answer geometric queries, such as the point location query. The incidence structure provides access to an ordered tree of logarithmic height containing the geodesics intersecting each edge. We first outline how the geometric information is added to the incidence structure, and then we show how to use this information to process a point location query.

For each directed edge we define a coordinate system. The origin of the system is the tail of the edge, the x-axis is directed along the edge, and the y-axis is placed on the plane of the adjacent face lying to the left of the edge. Each geodesic that traverses the edge can be represented as a linear equation with respect to this coordinate system. The geodesics in a bundle are implicitly represented with respect to the representative edge of the bundle. This is done as follows. For a single geodesic bundle (a leaf of the incidence structure), we store the equation of the geodesic explicitly as a linear equation $T(v)=c$ where T is a linear operator, c is a scalar and v is the vector indeterminate. Each link in the incidence structure is associated with a linear transformation converting from the child bundle's coordinate system to the parent bundle's coordinate system. Informally, by composing these transformations along the path from the root to the leaf and then applying T we can derive the equation of any geodesic with respect to any edge.

Specifically, let $b_1=(G_1, E_1)$ be a bundle in the incidence structure with a parent node $b_2=(G_2, E_2)$, and let $e_1 \in E_1$ and $e_2 \in E_2$ be the respective representative edges for these bundles. This implies that E_2 is a subsequence of E_1, and G_1 is a subsequence of G_2. Unfold the edges of E_1 so that their adjacent faces lie in a common plane. The geodesics in G_1 are mapped to straight lines after this unfolding. Note that $e_2 \in E_1$ is a part of this unfolding. There is a unique Euclidean transformation that converts a vector v represented in the coordinate system of e_2 to the equivalent point in the coordinate system of e_1. Call this transformation C_{b_1,b_2}, and label the link between b_1 and b_2 in the incidence structure with this transformation.

Letting $T(v) = c$ denote the equation of a generic geodesic in G_1 with respect to e_1, it is clear that the equation $T(C_{b_1,b_2}v) = c$ is the equation of this same geodesic in G_2 with respect to e_2. Thus, if $b_1, b_2, \ldots, b_k$ form a path in the incidence from a leaf to a root, where b_1 is a single geodesic bundle for a geodesic g and b_k is a single edge bundle for an edge $e \in I(g)$, then the equation for g in the coordinate system of e is given by composing these transformations:

$$T(C_{b_1,b_2}C_{b_2,b_3} \ldots C_{b_{k-1},b_k}v) = c.$$

This composition can be performed by matrix multiplication. Thus, the equation of any geodesic with respect to an edge that it traverses can be computed in logarithmic time by traversing the path from the root to the geodesic.

Left and right discriminations can also be made by replacing the equality $T(v) = c$ with an inequality $T(v) \geq c$ devised, say, so that the head of the edge satisfies the inequality. The marks on the incidence structure links are used to reverse the direction of the inequality. For every node in the incidence structure, we precompute the equation of the leftmost geodesic in the right subtree and the rightmost geodesic in the left subtree and store these equations in the node. This can be performed in time linear in the size of the incidence structure by a simple traversal of the structure. With these enhancements we can answer the following restricted version of the point location query.

Point Location on an Edge. Given a point x located on an edge e of the polyhedron, where x is represented in the coordinate system for e, determine the two geodesics or endpoints of e that immediately surround x on the edge.

The algorithm is recursive. At each level of the recursion we are given a bundle b and the point x represented with respect to the representative edge for b. Initially, b is the single edge bundle for the query edge e. The ouptut of each recursive invocation will either be a pair of geodesics directly bounding x or else an indication that x lies entirely to the right or left of the bundle, where the head of the representative edge is taken to be far right.

Algorithm 4.1. Locate x with respect to the bundle b.

1. If b is a single geodesic bundle, then return "left" or "right" depending on whether x lies on the same side of the geodesic as the tail or head, respectively, of the representative edge for b.
2. Otherwise, b has two children. Let T_1 and T_2 be the equations of the rightmost geodesic of the left subtree and the leftmost geodesic of the right

subtree of b, respectively. If x lies between T_1 and T_2, then return the names of the corresponding pair of geodesics.

3. Otherwise, x lies to the left of T_1 or to the right of T_2. In the former case, let C_1 be the transformation on the link to the left subtree of b. Recursively locate the point C_1x in the left subtree of b. If the result of the recursive call is "left" or "right" and the link between b and the left subtree is marked, then reverse the direction of the result before returning. A symmetric operation holds in the case of searching the right subtree.

The correctness of the algorithm follows from the preceding discussion. The invariant regarding the relative order of intersection of the geodesics on the representative edges follows from the fact that geodesics do not cross. The search time is proportional to the depth of the incidence structure, $O(\log(n+m))$. Note that once a pair of geodesics (or endpoints of e) that bound a query point x are known, the region of the subdivision containing x is determined. To compute the region, we store each pair of adjacent geodesics in a dictionary with a fast search procedure to look up the region of the subdivision. The number of adjacent geodesics is bounded by the size of D, the set of double geodesic bundles. Thus, by Lemma 3.3, the size of this dictionary is $O(n+m)$.

At this point we have developed essentially enough mechanism to solve the general point location problem.

Polyhedral Surface Point Location. Given a subdivision Q partitioning the surface of a polyhedron into regions bounded by geodesics, and given a query point x on the surface of the polyhedron, find the region of Q that contains x.

The query consists of a face f and the coordinates of a point x on f with respect to f's coordinate system. Some preprocessing of the polyhedron is required, prior to building the incidence structure. Consider the face f, and let S denote the set of subdivision vertices (geodesic endpoints) lying in the interior of f. Triangulate f together with the vertices S, forming a refined polyhedron with possibly coplanar faces. This triangulation forms a planar polygonal subdivision of f; see Fig. 9. The size of the triangulated polyhedron is $O(n+m)$ by its planarity and Euler's formula, and it can be computed in

$$O((n+m)\log(n+m))$$

time by standard techniques [14]. Using standard point location methods, we can determine which refined face contains the query point x in logarithmic time and overall $O(n+m)$ space [4], [7].

Next, we build the incidence structure for the refined polyhedron. Consider the triangulated face f of the refined polyhedron that contains the query point. Because of our refinement, there are no subdivision vertices in the interior of this face. Since the restriction of a geodesic to a face is a straight line, it follows that every subdivision region intersecting f either contains f or else transversely intersects at least one of the edges of f. In the former case, knowing the face that contains the query point x determines the region of the subdivision. Thus, we

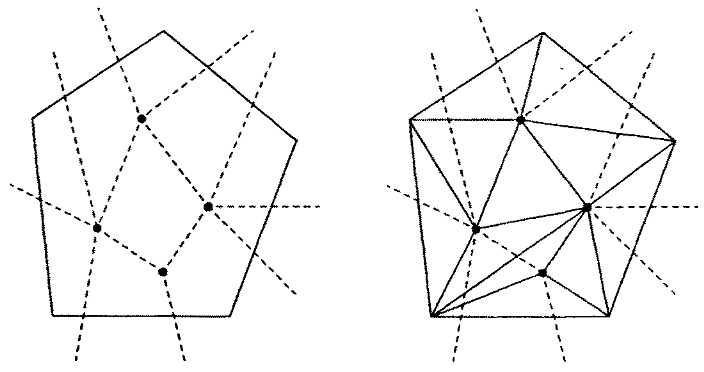

Fig. 9. Triangulating of a polyhedron face.

concentrate on the latter case. It suffices to determine two geodesics bounding the region containing x that are adjacent on some polyhedron edge. We have developed the capability of performing a tree search on the geodesics crossing any given edge of the polyhedron. Intuitively, to solve the point location problem we repeat this operation on every edge surrounding the fact that contains x.

Consider an arbitrary geodesic g, and assume that g traverses an edge e of the face. Recall that $J(e)$ denotes the set of all geodesics traversing e. If g does not intersect the interior of the face, then g must be incident on an endpoint of e, say the tail v of e. In this case g and the subset of $J(e)$ lying to the left of g (clockwise about v) can be eliminated from the search. A symmetric argument holds if g is incident on the head of e. If, on the other hand, g traverses the interior of f, then, because there are no subdivision vertices on the interior of f, g subdivides f into two regions such that all the geodesics to the right of g in $J(e)$ lie entirely in one region, and the geodesics to the left of g in $J(e)$ lie in the other region. Thus, by applying a tree search to the subtree of the incidence structure rooted at e, we can discriminate which two geodesics of $J(e)$ bound the query point x in logarithmic time. In general, x may be bounded by only one geodesic, but we ignore this case for the sake of simplifying the description.

We repeat the discrimination on each of the three edges of f. We claim that these three discriminations uniquely determine the region of the subdivision that contains x. The following cases arise:

1. If for two edges of f we find the same pair of geodesics bounding x, then it follows that these two geodesics bound the region containing x in the subdivision.
2. If for all three edges of f we find a different pair of geodesics bounding x, it follows that there are at least three geodesics bounding x in f. However, there can be no more than three geodesics bounding x, because each geodesic must intersect an edge of f, and f is a triangle. It follows that all three geodesics bound the region of the subdivision that contains x.

In either case, we have determined a pair of geodesics that are adjacent on some polyhedron edge that bound the region containing x. The region is determined by consulting the dictionary of adjacent geodesic pairs. As mentioned earlier, the size of the dictionary is bounded by the size of D, the set of double geodesic bundles, which was shown to be $O(n+m)$ by Lemma 3.3.

This completes the description of the solution to the point location query. The complexity is $O(\log(n+m))$ since three tree searches are performed in the incidence structure, whose height is $O(\log(n+m))$.

5. Further Remarks

We have described a data structure, called the incidence structure, based on a collection of trees with shared subtrees that records the ordered incidence of a collection of geodesics on a polyhedron's surface. The structure consists of a discrete component, which depends only on the order in which geodesics intersect polyhedron edges, and a geometric component, from which the exact equation of each geodesic on any face can be determined. We have shown how to answer a number of natural queries based on simple tree traversals. The structure can be viewed as an unoriented counterpart to geometric search tree structures based on global orientation such as plane sweep [3], [13] or discrimination with respect to monotonic chains [4], [8]. The inadequacy of existing geometric search strategies for our problem stems from the nonmonotonic nature of geodesics.

There are a number of interesting issues that are raised by this work. The discrete component of the data structure can be defined in any situation in which two planar graphs are overlaid. Guibas and Seidel have considered quite a different problem in this same setting [5], although the techniques that they apply are quite different from ours. Applications of this structure to other areas in graph theory and discrete geometry may exist.

We have presented one method of constructing the incidence structure by performing full ties selected as an independent set from a planar graph. Although our method gives asymptotic bounds, it is doubtful whether this method is of direct practical value. It would be of interest to know if there are simpler, more direct methods that guarantee the same performance bounds, or whether random tying or random full tying will yield as good a performance on the average.

One application of this structure is in solving the nearest-neighbor problem on the surface of a convex polyhedron. Generalizing this result to nonconvex polyhedra requires allowing curves that are "hyperbolic geodesics." Although the incidence structure can easily represent such curves, the search procedure of Section 3 requires that no geodesic cross the same edge twice. This may be violated by hyperbolic geodesics. Generalizing the search procedure to this case would be of interest. Furthermore, it would be interesting to know whether the incidence structure can be built "on the fly" while constructing the Voronoi diagram on the surface of the polyhedron, thus avoiding the $O(nm)$ space requirements of both algorithms [9], [11]. Although our approach seems to require $O(nm)$ time and space just to store the input, there may be quite different approaches that obviate the need for maintaining all of this information.

Acknowledgments

The author would like to thank Azriel Rosenfeld for his comments on an earlier draft of this paper, to Deepak Sherlaker for providing a relevant reference, and to the referees for their comments, which improved the presentation of the paper.

References

1. N. Chiba, T. Nishizeki, and N. Saito, A linear 5-coloring algorithm of planar graphs, *J. Algorithms* **2** (1981), 317-327.
2. D. Dobkin and D. Kirkpatrick, Fast detection of polyhedral intersections, *Theoret. Comput. Sci.* **27** (1983), 241-253.
3. D. P. Dobkin and J. I. Munro, Efficient uses of the past, *J. Algorithms* **6** (1985), 455-465.
4. H. Edelsbrunner, L. J. Guibas, and J. Stolfi, Optimal point location in a montone subdivision, *SIAM J. Comput.* **15** (1986), 317-340.
5. L. Guibas and R. Seidel, Computing convolutions by reciprocal search, *Proceedings of the Second Annual ACM Symposium on Computational Geometry*, 90-99, Yorktown Heights, New York, 1986.
6. L. Guibas and J. Stolfi, Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams, *ACM Trans. Graphics* **4** (1985), 74-123.
7. D. G. Kirkpatrick, Optimal search in planar subdivisions, *SIAM J. Comput.* **12** (1983), 28-35.
8. D. T. Lee and F. P. Preparata, Location of a point in a planar subdivision and its applications, *SIAM J. Comput.* **6** (1977), 595-606.
9. J. S. B. Mitchell, D. M. Mount, and C. G. Papdimitriou, The discrete geodesic problem, *SIAM J. Comput.*, to appear.
10. D. M. Mount, On Finding Shortest Paths on Convex Polyhedra, Technical Report 1495, University of Maryland, 1985.
11. D. M. Mount, Voronoi Diagrams on the Surface of a Polyhedron, Technical Report 1496, University of Maryland, 1985.
12. D. E. Muller and F. P. Preparata, Finding the intersection of two convex polyhedra, *Theoret. Comput. Sci.* **7** (1978), 217-236.
13. F. P. Preparata, A new approach to planar point location, *SIAM J. Comput.* **10** (1981), 473-482.
14. M. I. Shamos and D. Hoey, Closest-point problems, *Proceedings of the 16th IEEE Foundations of Computer Science Symposium*, 151-162, 1975.
15. M. Sharir, Intersection and closest-pair problems for a set of planar discs, *SIAM J. Comput.* **14** (1985), 448-468.
16. M. Sharir and A. Schorr, On shortest paths in polyhedral spaces, *SIAM J. Comput.* **15** (1986), 193-215.

Received April, 1986, and in revised form December 3, 1986.