Efficient Computation of Geodesic Shortest Paths

Sanjiv Kapoor
Department of Computer Science
Indian Institute of Technology
Hauz Khas, New Delhi 110016

May 17, 1999

Abstract

This paper describes an efficient algorithm for the geodesic shortest path problem, i.e. the problem of finding shortest paths between pairs of points on the surface of a 3-dimensional polyhedron such that the path is constrained to lie on the surface of the polyhedron. We use the wavefront method and show an $O(nlog^2n)$ time bound for this problem, when there are $O(n)$ vertices and edges on the polyhedron.

1 Introduction

Terrain Navigation and Path Planning are important poblems in Robot Motion Planning. Both 2-d and 3-d Path Planning problems have been well studied. While efficient solutions to the 2-d problem are well known, the 3-d problem is notoriously hard. The problem of determining the shortest path between two points in 3-d amongst polyhedral obstacles is NP-Hard. More success has been achieved with scenes of less complexity.

In this paper we consider the *Discrete Geodesic Problem*, that is the problem of finding shortest paths between pairs of points on the surface of a 3-dimensional polyhedron such that the path, termed *geodesic path*, is constrained to lie on the surface of the polyhedron. We let n be the number of vertices of the polyhedron. And we let e be the number of edges. W.l.o.g we assume that each face of the polyhedron is a triangle. The complexity of the description of the scene will be taken as $O(n)$.

The problem is of obvious interest in terrain navigation. This problem was first posed by Sharir and Schorr [SS] who provided an $O(n^3logn)$ solution for the case when the polyhedron is convex. An improved solution provided by Mount [Mo] has complexity $O(n^2logn)$.

For non-convex polyhedron the first algorithm presented was of $O(n^5)$ complexity [RSB]. The time complexity was subsequently improved by Mitchell, Mount and Papadimitrou [MMP]. In their algorithm a shortest path is evaluated by maintaining intervals of optimality on each edge. The non-discrete nature of the intervals provides the algorithm the name *continuous Dijkstra*.

In this paper we employ a methodology which is even less discretized and has come to be termed the *waveform propagation* method. The waveform propagation method has been used in shortest path computations in a number of papers [Mi2]. We show how to efficiently design a propagation method to compute shortest paths on the surface of a polyhedron. The time complexity of this method is $O(nlog^2n)$. There are three essential components of the algorithm.

- The shortest geodesic waveform. This waveform essentially comprises sequences of arcs of circles centered at vertices of the polyhedron called source vertices. The waveform is split into sections, each section being associated with an edge.
- An unfolding of the surfaces of the polyhedron for propagation of the waveform. A surface may occur with different orientations in the distinct unfoldings. For efficient maintenance of such an unfolding structure we employ a method of alignment of waveform sections associated with an edge in the different unfoldings.
- A convex representation of the waveform for efficient shortest distance computations. This representation has been used before in [KM2]. The manipulation of the convex representation is similar to that in [KM2]. We borrow freely from that work.

2 Preliminaries and Definitions

Let $\mathcal{P}$ be a polyhedral surface in R^3 and let s and t be two points on the surface of $\mathcal{P}$. $\mathcal{P}$ is specified by faces which are closed polygons. The faces intersect at edges and vertices. W.l.o.g assume that the faces are triangles. s and t are referred to as source and destination, respectively.

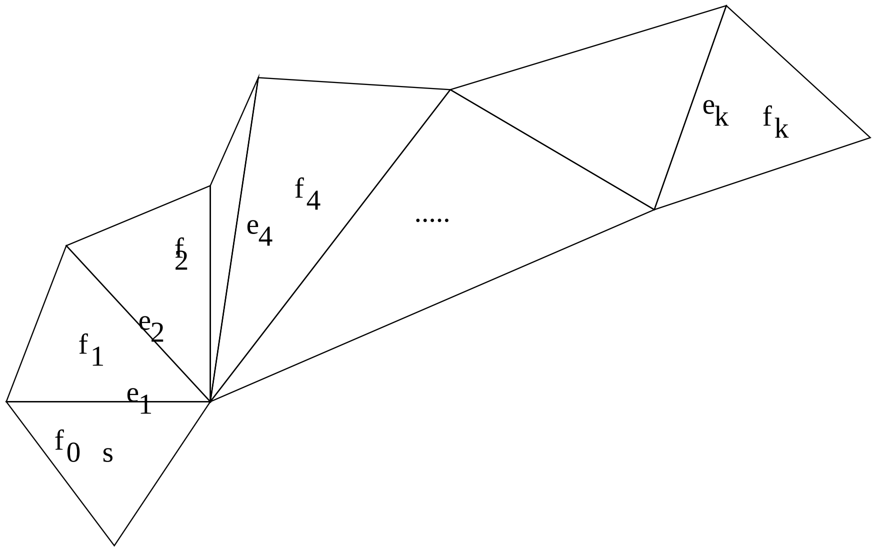

Figure 1: Unfolding of faces

The *Discrete Geodesic Problem* is to find the shortest path in the Euclidean Metric from s to t such that the shortest path is constrained to lie on the surface of P.

We next present definitions. We borrow terminology from [MMP]. Two faces f_1 and f_2 are said to be *edge-adjacent* if they share a common edge e. $\mathcal{F} = \{f_1, f_2 \dots f_k)$ is a sequence of edge-adjacent faces, called a *face-sequence*, such that f_i is edge-adjacent to f_{i+1} with e_i as the edge common to f_i and f_{i+1}. The root of the sequence is the vertex in f_1 not the endpoint of e_1. It is referred to as $root(\mathcal{F})$.

Suppose f and f' share an edge e. We define the planar unfolding of face f' onto face f as the image of f', $Im_f(f')$, when rotated about the line containing e onto the plane containing f such that $f \cap Im_f(f') = e$. The planar unfolding of the sequence $\mathcal{F}$ is the planar unfolding of $f_2, f_3 \dots f_k$ onto the face f_1. We shall refer to the unfolded image of an edge, e with respect to the planar unfolding of the sequence, $\mathcal{F}$, as $U(\mathcal{F}, e)$.

A *path* on $\mathcal{P}$ is a simple geodesic path if it is simple and cannot be shortened by slight perturbations. Furthermore, path P connects an edge sequence E if P comprises segments that join interior points of E. We can state the following lemmas about geodesic paths:

Lemma 2.1 *If P is a geodesic path which connects the edge-sequence E then the unfolded image of P along the edge sequence E is a straight line segment.*

Lemma 2.2 *[MMP] The general form of a geodesic path is a path which goes through an alternating sequence of vertices and edge sequences such that the unfolded image of the path along any edge seqence 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.*

2.1 Algorithm Outline

We need the following definitions specific to our algorithm: Assume that s lies on face f_0. The *GSP-Waveform* (Geodesic Shortest Path Waveform) is the locus of points at shortest geodesic distance d from the source vertex s.

Initially, the GSP-Waveform is a circle of radius ϵ centered at s. In general, the GSP-Waveform is a sequence of arcs of circles with centers as vertices of the polyhedron.

The algorithm proceeeds by expanding the waveform starting with the initial waveform. The waveform expansion is over faces unfolded and is guided by event points. During the expansion the waveform arcs may change shape, new arcs may be incorporated into the waveform and arcs removed. Also non-adjacent arcs in the waveform can also intersect. The waveform, in general, comprises disjoint cycles which enclose regions to which the shortest path has not been determined. We need only consider the waveform cycle enclosing the region which contains the destination vertex.

A face of the polyhedron is defined to be completely traversed if every edge of the face has been crossed by a section of the waveform from inside the face, i.e. there is a point on the edge to which the shortest geodesic path traverses the inside of the face. A face is partially traversed if there is an edge of the face that has not been crossed by a section of the wavefront. Partially traversed faces are useful because the untraversed bounding edges and vertices of the face give rise to event points. We will call partially traversed faces *PT-faces*.

Let f be a face. A face-sequence , SF, unfolded onto the plane of f defines a region which we term $UR(SF, f)$. Each face-sequence when unfolded forms a connected region. We let $\mathcal{US}$ be the union of all unfolded face-sequences. Note that $\mathcal{US}$ may comprise overlapping regions. Each edge on the boundary of $\mathcal{US}$ is termed as a *boundary edge*. With the plane containing f, is associated a co-ordinate system.

An edge e of a face f_1 may occur in two different unfoldings of face-sequences containing f_1. We denote by $Ang(U(\mathcal{F}, e))$, the slope of the edge e when the face-sequence $\mathcal{F}$ is unfolded onto f. And we denote by $Ang(e, \mathcal{F}_1, \mathcal{F}_2)$ the change in orientation required to align edge e in the two unfoldings $\mathcal{F}_1$ and $\mathcal{F}_2$. For this purpose, we assume a common co-ordinate system for the various unfoldings.

With each PT-face we associate a section of the wavefront. These are the sets of arcs such that an arc in this set is a candidate for striking an untraversed edge of the face. For simplicity we will assume that each face is a triangle. An arc may actually be associated with a number of edges of different faces. For efficiency purposes these are clubbed into one sequence termed *boundary-section*. The edges are sequenced by their ordering around the center of the arc.

The following invariants will be maintained in the algorithm:

(i) An unfolded structure, $\mathcal{US}$, obtained from $UR(SF, f_0)$, i.e. sequences of edges unfolded onto the initial face.

(ii) An association between boundary edges and wavefront arcs. A boundary edge is associated with one or many arcs and a wavefront arc is associated with one or many boundary edges. As will be shown the boundary edges associated with one arc form a sequence if they lie on the boundary of an unfolded face-sequence. The arcs associated with a boundary edge are combined into a sequence of arcs. Since an edge may occur more than once in the unfolded structure, the various wavefront arcs associated with the edge are incorporated by merging the waveform sections after rotating them so that the corresponding edge occurences are aligned. Note that when an arc of the wavefront is associated with more than one edge at different angular orientation then the arc has been split into parts. The number of such splits is bounded.

(iii) Hierarchical Convex hull structures with every sequence of boundary edges and wavefront arcs. The convex structures are useful in creating *approximate* boundary and wavefront sequences. The convexity aids in computing the shortest distances between the boundary edges and wavefront arcs.

The initial waveform is a circle centered at the start point. The initial unfolded structure is the face f_0. The waveform expands until it strikes either an edge of the boundary or a vertex. The following event points are important:

- **Edge Strikes**: This occurs when an edge of the polyhedron is closest to some arc center s than any other untraversed edge or vertex. The waveform when expanded strikes the edge. Let e be the edge struck. e exists on the unfolded stucture $\mathcal{US}$, when a triangle T_1 is unfolded onto f_0. At this point the other triangle with edge e as one of its edges is unfolded onto f_0 and the other two triangle edges are brought into the boundary of the unfolded face-sequence. The section of the waveform to be associated with edges of T (apart from e) are determined. There are two cases:

  (1) Triangle has not been traversed before. In this case the association determined is confirmed.

  (2) Triangle has been traversed. In this case a merge step of the new waveform arcs with the already existing arcs associated with edges of the triangle is required.

- **Vertex Strike** If the waveform strikes a vertex, v, then the shortest path to v has been determined and v is labelled. Consider all faces with v as a vertex and with a point p such that the angle formed by (p, v) and (v, s) is greater then π. If f is such a face then an unfolded face-sequence is started using f with a waveform arc on it with v as root. The initial waveform for the face f is an arc with v as center. Merges of the edges of the boundaries of f are done if required.

Note the importance of the convex hull representation in determining the above strikes. Naively, $O(n)$ arcs may strike a boundary edge and the first arc to strike a boundary edge is to be determined from these arcs. The convex hull representation which approximates the shape of the arcs associated with a boundary edge allows efficient determination of the strike. Details are in the following sections.

Other events that change the GSP-Waveform are removals of an arc or a set of arcs. Again these are efficiently managed by convex representations of the waveform and boundary-sections.

All events are scheduled using a heap.

We describe details below.

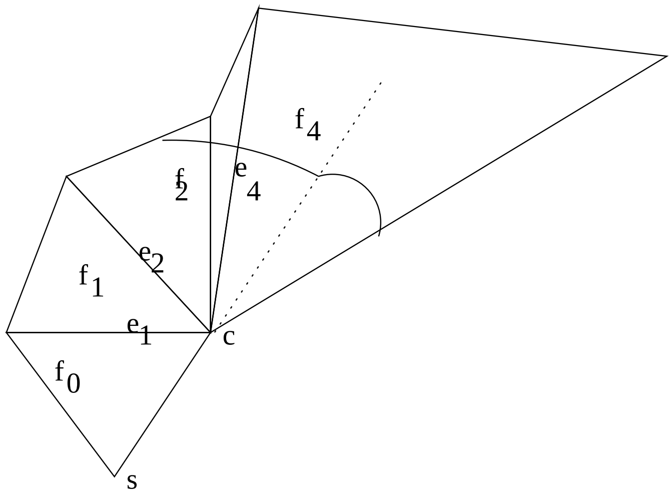

Figure 2: GSP-Waveform on an unfolded sequence of faces

2.2 Waveform Characterstics and Distance Computations

In what follows we consider the waveform restricted to an unfolded face or a face-sequence.

We re-iterate the following definitions: The *GSP-Waveform* (Geodesic Shortest Path Waveform) is the locus of points at shortest geodesic distance d from the source vertex s. A face is partially traversed if all the edges of a face have not been crossed (from the face) by a section of the wavefront. Such faces of the polyhedron are called PT-faces.

We characterize the waveform next. The GSP-Waveform at geodesic distance d from the source will be characterized by the shortest paths themselves. The shortest path is either a straight line path to the source on an unfolded face-sequence or a straight line to some other vertex of the scene. We will partition the waveform into sets of points so that points in a set have the same vertex as the first vertex in the shortest geodesic path to the source. This results in a partition of the waveform into a sequence of circular arcs $A_1, A_2 \ldots A_k, k \leq n$ such that all points in A_i have a vertex v_i as the first vertex in the shortest path to the source. The arc A_i has center v_i and radius $d - d_{vi}$ where d_{vi} is the shortest geodesic distance from v_i to the source vertex. Note that the source vertices are the vertices of the polyhedron.

Furthermore, the locus of points which are the common end-points of two adjacent arcs A_i and A_{i+1} of successive waveforms is called an I-curve denoted by $I(A_i, A_{i+1})$. This curve is the locii of points at the same geodesic distance from the source vertex using the two centers v_i and v_{i+1} as the last vertices on the shortest path.

We next show in detail how to compute the distance to a boundary given the GSP-Waveform.

The relevant boundary edges to which distances are required to be computed, are obtained from PT-faces. Let $\mathcal{T}$ be the set of partially-traversed faces in $\mathcal{US}$. Consider a region $UR(SF, f)$ and the GSP-Waveform, W_f, restricted to that region. The GSP-Waveform is a simple connected curve. We let C_f be the vertices of the polyhedron that are centers of arcs in W_f.

Suppose we traverse the GSP-Waveform W_f starting from one end. A GSP-Waveform arc *crosses* an edge of a triangle, T, when the arc intersects the edge. Let FNE be the set of edges which are not crossed by the GSP-Waveform and are on the boundary of partially-traversed face in $UR(SF, F)$. And let $T_1, T_2 \ldots T_k$ be the partially traversed faces encountered by the GSP-Waveform during the traversal from one end to the other. Each partially-traversed face *contributes* an edge or two consecutive edges to FNE. The sequence of edges on the boundary of $UR(SF, F)$ is obtained from edges in FNE contributed by $T_i, 1 \leq i \leq k$.

We next show how to associate waveform arcs with triangle boundary edges. The association of waveform arcs with boundary edges, as required, is as follows: Let A be a waveform arc such that A is defined by a vertex v in the polyhedral scene. Let e be a boundary edge contributed by a triangle T. e is *associated* with A if points on e are closest to A than to any other arc in W_f. Note that a boundary edge can be associated with more than one waveform arc. Also, more than one boundary edge can be associated with an arc.

Contiguous vertices and edges of the boundary of an unfolded face-sequence associated with arc A, form a section called *boundary-section* and this boundary-section is associated with the arc A. More than one such section can be associated with A.

An arc may be only associated with one boundary edge e in a triangle T. Firstly, all the contiguous arcs associated with the triangle edge form a waveform-section. This waveform-section is associated with the edge of the triangle. Moreover, edge e may be associated with a number of waveform-sections which strike it after crossing a particular edge, say e' of the face T. All these waveform sections associated with an edge are combined into one section after aligning the occurence of edge e' in the set of unfoldings. The waveform section are combined in the order in which the arcs, associated with the edge e', occur in

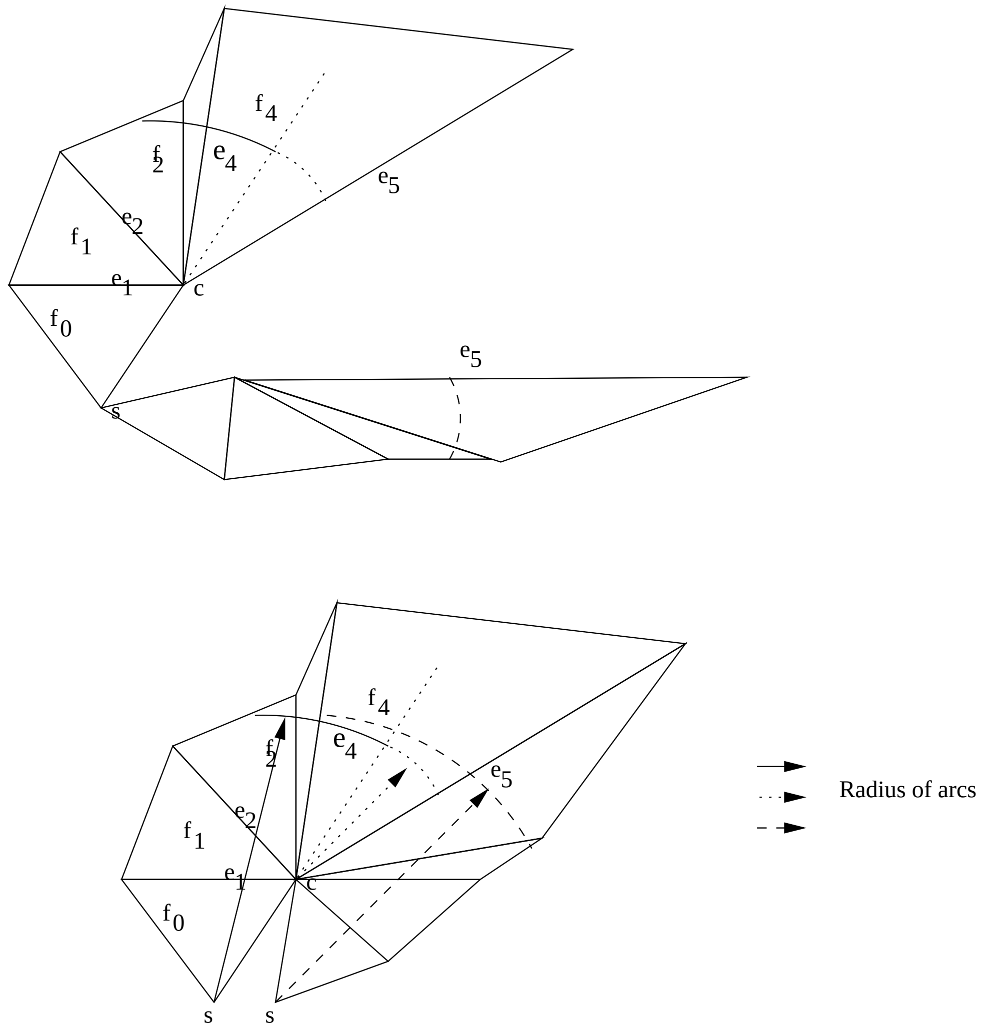

Figure 3: Aligning edges in different unfoldings

the waveform. A triangle thus has a constant number of waveform sections associated with it, each one associated with the edges of the triangle which occur on the boundary.

Furthermore, we consider some properties of the boundary-sections associated with an arc A.

We let $RVF(A)$ be the set of all points on the currently unfolded faces w.r.t. the face containing A and which are closer to arc A than to other arcs. The boundary of $RVF(A)$ comprises boundary edges of the unfolded faces and $I(A, X)$, I-curves between A and another waveform segment X. The center of this arc segment may lie outside the unfolded sequence of faces containing points associated with A. X is called a neighbour of A. Note that the boundary of $RVF(A), \forall A$ is linear in the number of vertices since the regions partition the planar unfolded sequences.

However, during the course of the algorithm we will restrict our attention to one boundary cycle comprising edges associated with a given arc and hence to one contiguous sequence of boundary edges associated with an arc. Consider boundary edges which have not been struck from any one of the two faces they border. It is easy to see, by induction, that these boundary edges form a set of cycles on the surface of the polyhedron. Consider the cycles containing edges associated with the given arc. We will restrict our attention to the cycle which encloses the destination vertex. We let $S_1 = RV(A, \delta)$ be a sequence of edges on a boundary cycle, δ, associated only with A, ordered by the angle that the shortest path from points on the edges makes at the center of A. The associations satisfy the following contiguity property:

Contiguity Property: There exists an association such that S_1 is a contiguous sequence along the boundary.

We will let $RV(A)$ denote $RV(A, \delta)$ whenever δ is understood.

Finally, we take care of arcs in the waveform that are not associated with any edge of the boundary. During the course of the algorithm these arcs may be included within a waveform-section. These arcs are detected when two I-curves, say I_i and I_j, defined by arcs in a waveform section intersect within a triangle. The arcs, $A_{i+1} \dots A_j$, are said to *disappear* from the GSP-Waveform. We incorporate these

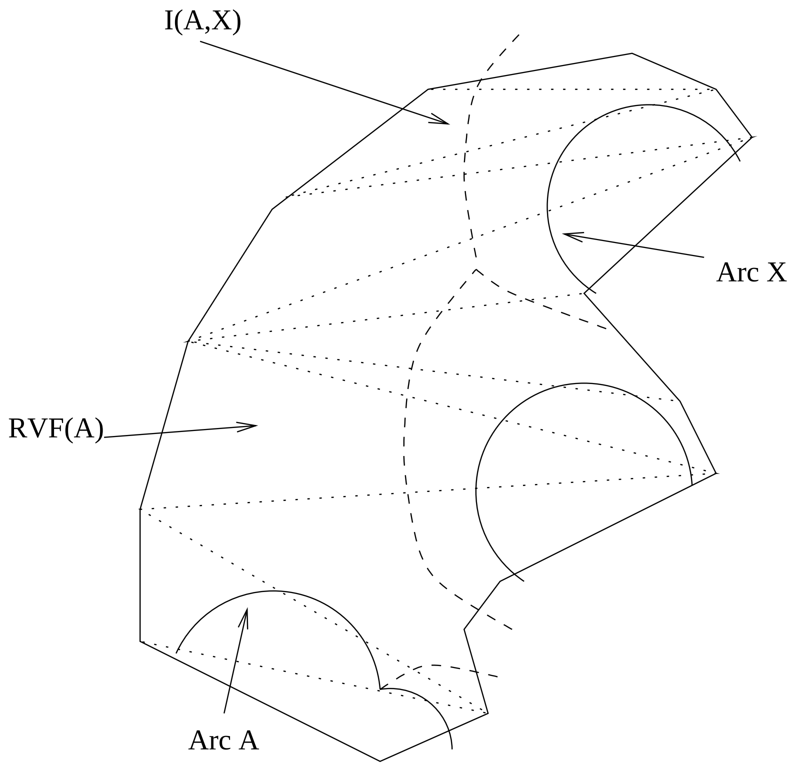

Figure 4: RVF(A)

arcs into waveform-sections which are formally defined as follows: A *Waveform-section* associated with an edge e in a triangle T, comprises sequences of arcs, $S_1 \ldots S_k$, such that a sequence comprises arcs $A_{i1} \ldots A_{ip}$ with each arc being either associated with e or being one which disappears in T.

The above associations of arcs to boundary edges are used to partition the boundary into boundary-sections. Each section of the partition is of one of the following types:
(1) A sequence of edges or vertices associated with an arc or a waveform-section.
(2) An edge associated with an arc or a set of arcs. The set of arcs is ordered in the sequence that the arcs occur along the wavefront.

We call these sections Type 1 and Type 2 sections respectively.
The following lemma bounds the size of the boundary-sections.

Lemma 2.3 *The number of vertices and edges in all the boundary-sections is $O(n)$.*

Proof: There are $O(n)$ boundary edges and vertices in ∂B since each edge is used in at most two directions. Each edge is part of one section. As we scan the flow wavefront an edge may repeat on the boundary being associated with a number of flow segments. However, the waveform segments associated with it are combined into one section when they cross a particular edge of the triangle containing that edge. The bound follows. ■

Also it is easy to see that

Lemma 2.4 *The number of Waveform-Sections is $O(n)$.*

Having computed the Boundary and Waveform-sections we next describe how to compute the distances of the waveform to the boundary-sections.

2.2.1 Boundary Distances and Section Representations

In this sub-section we show how to compute distances to boundary edges. This will require a convex representation of the GSP-Waveform and boundary-sections. We describe this representation also here.

The distance computations are between the waveform and boundary-sections.
(1) For Type 1 sections the distance is between the associated arc and an edge or a vertex of the edge on the boundary-section.
(2) For Type 2 sections it is the minimum distance between the waveform-section and the edge.

In our algorithm, shortest distances are efficiently computed by using the convex hulls of the waveform and the boundary-sections.

Convex Representations

Let W_s be a waveform-section with A_1 as the first arc and A_q as the last arc in the waveform-section. Suppose this waveform-section is currently on a sequence of unfolded faces UR_1 and the shortest distance from points on the wavefront to the source point is d. Let C_s be the vertices of the polyhedra that contribute arcs to W_s. Let u_1 be the starting endpoint of A_1 in W_s and let u_q be the last endpoint of A_q in W_s. Furthermore, let R_s be the region on UR_1 such that every point is at shortest distance less than d from s and uses a vertex or edge in C_s as the first vertex in $SP(p)$, the shortest path from p to the source vertex s. The boundary of R_s comprises W_s and another boundary, say O_s. W.l.o.g. we assume that R_s is a connected region. If not then, we can consider each of the connected regions independantly.

We need the following definitions: A non-intersecting curve, $C \in R^2$, is called *convex w.r.t. a point* p if for any pair of points r and s on C the triangle (p, r, s) lies inside the region formed by (p, r), (p, s) and the curve between r and s.

The *convex hull of a waveform section*, W_s, denoted by $CH(w_s)$ is defined to be the smallest convex piecewise linear or circular curve w.r.t. points on W_s that together with O_s encloses R_s. The arcs are portions of arcs in the waveform and the straight lines are tangents between two arcs of the waveform-section W_s. The convex hull essentially replaces W_s. It is easy to see that if the waveform-section strikes a boundary-section then so does the convex hull of the waveform-section. It thus suffices to restrict our attention to convex hulls of waveform-sections. A succesion of convex hull strikes will give the actual point of strike between a waveform arc and a boundary vertex or edge. The convex representation of the waveform-section is called a $C - Waveform\ Section$.

For efficient computation of shortest distances, the hull is stored in a balanced tree structure, $CWT(W_s)$, where every node of the tree stores the convex hull of the waveform section comprising arcs in the corresponding subtree. Also stored at each internal node v is an angle $Ang(v)$ which represents the angle $Ang(e, \mathcal{F}_1, \mathcal{F}_2)$ by which the waveform on an unfolded face-sequence, say $\mathcal{F}_1$, is required to be rotated, if required, so as to align with the sibling waveform which is associated with edge e on the unfolded face-sequence $\mathcal{F}_2$. Here e is the edge to be aligned during the merger of unfolded faces. The rotation required to align an arc at a leaf node, l, so as to obtain the composite waveform at the root node, is obtained from the rotations along the boundary from the leaf to the root node.

The leaves store the waveform arcs in the order that they occur in W_s. The convex hull at a node v of the tree is computed from the convex hulls at its sons. The bridge constructed at a node is either a tangent common to two arcs or a tangent from the endpoint of an arc to the other arc. The bridge construction procedure is as follows:

Procedure Bridge:

Let C_l and C_r be the convex hulls at the sons of node v. The bridge is constructed between C_l and C_r by using binary search on the hulls. There are two cases depending on whether C_l and C_r intersect or not. We show below that the convex hulls intersect at exactly one point if at all. Let $C_l = l_1 \dots l_p$ and $C_r = r_1 \dots r_q$. Let I be the intersection point of C_l and C_r. Then $I \dots l_p$ is contained in C_r and $r_1 \dots I$ is contained within C_l. To compute the bridges, firstly, the intersection point, I, of the hulls is found if it exists and then the bridge is constructed using the portions not included within, i.e. using $\{l_1 \dots I\}$ and $\{I \dots r_q\}$. Alternatively, if no intersection point exists then the bridge is constructed between C_l and C_r. The bridge construction procedure is the same as that of constructing bridges when the hull of a set of points is to be maintained [OL].

Lemma 2.5 *The Bridge Construction Procedure correctly computes the convex hull of the waveform in the subtree rooted at $v, v \in CWT(W_s)$ in $O(\log n)$ steps.*

Proof: The proof is by induction on the height of CWT. The base case is trivially true since the hull of an arc at a leaf node is the arc itself. Suppose we are given the correctly computed hulls at the sons of node v when it is not a leaf node. First, note that the arcs on the hull occur in the same order of occurrence as in the GSP-Waveform-section. We next show that there is exactly one intersection point between the the two hulls C_l and C_r.

The proof is by contradiction. Suppose there are at least two intersection points. Then there exists three points a, b and c in order along the hull such that the a and c belong to one sequence and b to the other. But this implies that the sequences intersect which is not possible.

Thus having computed the intersection point the bridge construction follows the procedure given in [OL] and its correctness follows. The convex hull at v is thus correctly computed. ∎

Thus:

Lemma 2.6 *C-Waveform-Sections can be constructed in $O(k \log n)$ operations where k is the number of arcs or line arcs in the waveform-section.*

With every bridge, B, in the convex hull tree is associated two curves, I-curves, I_{B1} and I_{B2}. Suppose the bridge B is constructed between arcs A_1 and A_2. I_{B1} is the locus of points equidistant from the bridge and arc A_1, and I_{B2} is the locus of points equidistant from B and A_2. The I-curve separating the bridge from an arc, say A_1, is normal to the bridge since the bridge is tangential to A_1.

In the distance computations we also need to compute convex hulls of the vertices of the boundary-section associated with a waveform arc, say A. The convex representation of the boundary-section $RV(A)$ is called a $C-Boundary\ Section$ and is stored in a tree structure, $CRV(A)$. The representation can be computed by a procedure similar to that which computed the convex representation of waveform-sections as described above.

Shortest Distance Computation
The detailed description is in the appendix.

We obtain the following result, where a *bridge split* is the removal of a bridge in the convex representation of a waveform section or a boundary section.

Lemma 2.7 *Given a $C-Waveform\ Section$ and a $C-Boundary\ Section$, the distance to the boundary-section can be computed correctly in $O(logn * (No.\ of\ Bridge\ splits))$ operations.*

Finally, we note that the assumption that no change takes place in the GSP-waveform is removed when we consider events that change the structure of the waveform. At these events the distance computations affected will be updated.

2.3 Maintaining Waveforms and Boundaries.

In the previous section we showed how to compute the distance to boundary-sections given the associated waveform arcs. This required computation of the convex hulls of the waveform-sections, C-waveform-sections, and the convex hulls of the vertices of the boundary-sections, C-Boundary sections. These can be computed initially as we scan the boundary.

However, the boundary and waveform-sections change as events are computed. At each change the distance to the current boundary needs to be updated. Thus one needs to maintain the boundary and waveform-sections and their convex representations dynamically since either arcs disappear from the waveform-sections or the waveform-section strikes a boundary thus changing the current boundary. In this section we show how to update the boundary and waveform-sections and the C-Waveform and C-Boundary sections after the change. The changes constitute event points. We first consider boundary changes. Next, we show how to handle changes in the GSP-Waveform and in the tree structures storing the C-Waveform and C-Boundary sections. Finally, we consider updating the event points themselves.

It is important to note that the boundary-sections and waveform-sections will be maintained modulo (i.e. subject to) changes in the event points.

2.3.1 Boundary changes

The boundary changes in two situations, once when a triangle edge is struck and the other when a vertex is struck.

- **Case A: Boundary edge strike:** When the boundary changes at the edge, $e = (v_1, v_2)$, a new triangle $T = (v_1, v_2, v_3)$ is unfolded onto the plane containing the current face-sequence. Suppose the triangle has not already been unfolded. The waveform-section to be associated with the new triangle is the waveform-section which strikes the edge. There are two sub-cases. Either the waveform-section striking the edge is a sequence of arcs, W, or a single arc, A. In the first case this sequence of arcs is split into two subsequences, W_1 and W_2, such that edge (v_1, v_3) is closer to arcs in W_1 and (v_3, v_2) is closer to arcs in W_2. The split portions are associated with the two new edges of the triangle in the boundary. In the second case the edges of the new triangle are added to the boundary-section associated with the arc. Finally, the distance to the new boundary can now be updated. In the first case this is done by computing shortest distances from W_1 and W_2 to the associated edges. In the second case the distance to the arc, A, from the boundary-section associated with it is recomputed.

  If the triangle has already been unfolded w.r.t. some other sequence of edges then the waveform-sections associated with the edges of the triangle are to be merged with the already existing sections in the triangle to update the waveform-sections associated with the new boundary edges. This may involve rotations of the waveform section striking the edge and a subsequent merge outlined below. The rotation is such that for both occurrences of the edge e, $Ang(U(\mathcal{F}_1, e)) = Ang(U(\mathcal{F}_2, e))$, where e belongs to two unfolded face-sequences, $\mathcal{F}_1$ and $\mathcal{F}_2$. We assume a common co-ordinate system.

- **Case B: Vertex strike** In this case, as outlined before, faces are unfolded if not already unfolded. A wavefront arc with v as center (source vertex) is generated for each unfolded face provided $\exists$ a point on the face which cannot be connected to a source vertex by a straight line on the unfolded face-sequence containing the face.

  If the face has not been unfolded before then the new waveform arc is associated with the edges on the boundary. If it has been unfolded before then a merger procedure is invoked as in the case of edge-strikes.

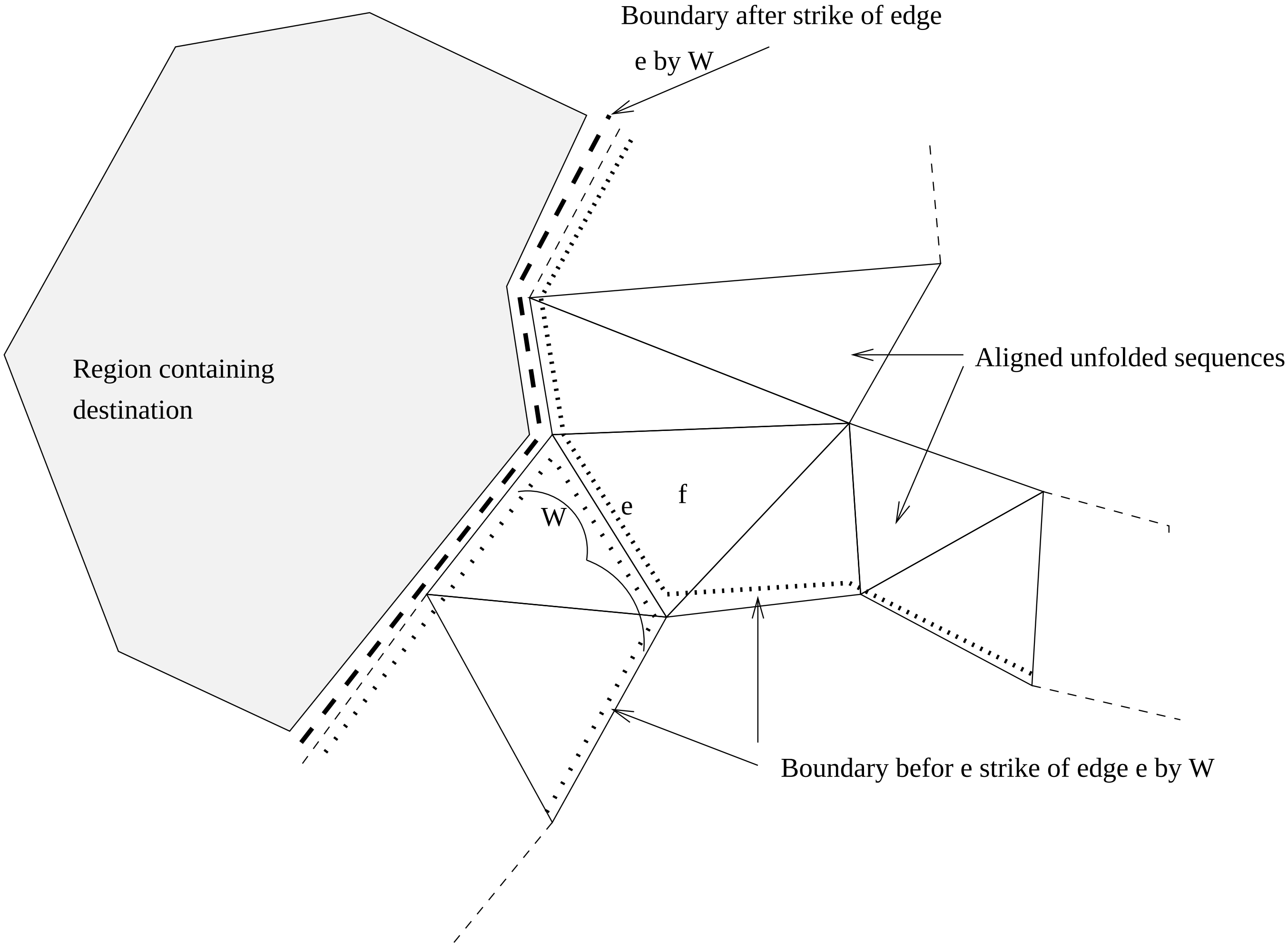

Figure 5: Merging after waveform W strikes edge e

2.3.2 The merging procedure

The merge procedure is used when the association of edges in the triangle $T = (p, q, r)$, and other relevant edges are to be changed due to a waveform-section, say W_n. Let $e = (p, q)$ be the boundary edge struck by the waveform W_n. We assume that there is only one waveform-section striking (p, q). If there are two, then they are combined into one. Note that there cannot be more than two waveform sections striking an edge. And let $e_1 = (p, r)$ and $e_2 = (q, r)$ be the other two edges of the triangle T. Moreover, let f_1 and f_2 be the two faces adjacent to T and which share edges e_1 and e_2 with T, respectively.

We need the following definition: Arc A *crosses edge* e *from face* f if point on f are closer to the center of arc A than any point on e.

Let e_1 and e_2 be the two new boundary edges which are to be associated with arcs in W_n. If e_1 has already been struck by a waveform arc or a waveform-section, we let W_{e1} be that section. W_{e1} is the waveform section that strikes e_1 after traversing face T. We assume that W_{e1} does not preclude W_n from crossing e_1. Similarly let W_{e2} be the waveform sequence that has already struck e_2. Note that if both W_{e1} and W_{e2} have struck their respective edges then one, say W_{e1}, will preclude W_n from crossing the other edge, which is e_2 in the case considered.

Because the triangle T has been previously visited, the merge will generate additional edge disjoint components in the set of components obtained from the boundary edges that have not been struck from any of the two faces they border. We need only consider processing one such component. The component we will process will have the destination vertex in it. This component is determined by the ordering of the unfolded sequences, $\mathcal{F}_1$ and $\mathcal{F}_2$, around the source vertex in a planar embedding of the graph corresponding to the edges of the polyhedron with the face containing the destination vertex being the unbounded face. This ordering identifies the boundary edge and vertices in the current triangle which border the component containing the destination.

We thus first update the boundary by splitting and merging the sequence of edges on the boundary. Next, the merger of the waveform sections is accomplished by one of the appropriate mergers depending

on the edge which may lead to changes of the arc associations with the boundary cycle under consideration:
(i) W_{n1} with W_{e1} where W_{n1} is the section of the waveform W_n associated with points in f_1 after crossing e_1.
(ii) W_{n2} with W_{e2} where W_{n2} is the section of the waveform W_n associated with points in f_2 after crossing e_2.

We detail merger (i). The other merger is similar.

We use the procedure $MODIFY(RV(A), W_{n1})$ given below.

We let B be an arc in W_{e1} associated with point p and the edge starting at p in the appropriate direction starting at the endpoint of e_1, p, on the boundary under consideration. We use the following procedure to modify $RV(B)$. Subsequently, if all edges in $RV(B)$ are associated with other waveform arcs then we modify recursively $RV(C)$ where arc C follows B in the waveform W_{e1}.

The modification, $MODIFY(A, W_{n1})$, requires associating edges in $RV(A)$ with arcs in W_{n1}. This is achieved as follows. Note that $CRV(A)$ is the convex representation of $RV(A)$.

Algorithm $MODIFY(RV(A), W)$
Begin
  Let $W = w_1 w_2 \ldots w_k$
  $i := 1$;
  Repeat
    Determine where I-curve $I(w_i, w_{i+1})$ intersects $RV(A)$
      using $I - intersect(I, CRV(A))$
    Let the intersection be at boundary edge e_a.
    Update $RV(w_i)$;
    Update $CRV(w_i)$;
    Find w_j where $j > i$ is the first index s.t. $I(w_{j-1}, w_j)$
      does not intersect e_a.
    $i := j$;
  Until $i = k$ or all edges in $RV(A)$ considered;
End.

Also, note that in the above procedure it may be possible that $RV(A)$ itself becomes empty. Then as stated before we modify $RV(B)$ where B follows A in W_{e1}.

In the above algorithm $MODIFY$ we use the following procedure where I is an I-curve between A_1 and A_2.:

Procedure $I - intersect(I, CP)$:
In this procedure, a binary search is used to search for the segment where I intersects the sequence of arcs or boundary section P. We assume that the convex representation of P, CP, is stored in CPT. We start at the root of CPT. At any node, v , let $B_v = (a, b)$ be the bridge at v. The procedure first determines if B_v is visible from c_1 (the center of A_1), i.e the line segment from c_1 to both endpoints does not intersect CP, taking into account the orientation of the arcs, if any, at that node. If not then the procedure branches to the left or right depending on which part of CP is visible. Alternatively, if a is closer to c_1 than b is, then the procedure branches to the left otherwise it branches to the right.

The above procedure requires $O(\log n)$ steps. Assume that $RV(A)$ does not become empty. The algorithm $MODIFY$ requires time $O((p+q)logn + logn)$ to update the sequences $RV(A)$ and $RV(w_i)$ where p is the number of flow-segments in W_{n1} which have a contiguous segment in RV associated only with the flow segment after invocation of $MODIFY$ and q is the number of splits of the contiguous sequence of edges associated with $RV(A)$ for some A. The splits involve change of associations of edges to arcs in $MODIFY$.

Furthermore, the path boundary is updated by splitting and merging two sections of the path boundary.

The case when the wavefront uses a vertex or edge of the triangle as a source vertex is similar to that above.

2.3.3 Correctness and Complexity of Boundary Changes

The following lemma follows from the above discussion:

Lemma 2.8 *MODIFY correctly updates $RV(A)$. Furthermore, all invocations of MODIFY in the algorithm require $O(nlogn + nU)$ steps, where U is the time to update $CRV(A)$ for some waveform arc A.*

Proof: The proof of correctness follows from the correctness of $I - intersect$ and the fact that any change in association when a merger, say W_{n1} with W_{e1}, takes place will involve change of association of A, the first arc in W_{e1}, followed by a change of association of arcs ocurring subsequently in W_{e1}. The change of associations of A are obtained from the various contiguous sections in $RVF(A)$. Other change of associations are obtained recursively from associations of arcs which are neighbors of A. It suffices to consider these contiguous arcs since associations on a boundary are contiguous.

We consider the time bound in detail. It suffices to bound p and q throughout the course of the algorithm. First consider the bound on p. We show that the total number of additions to this set is $O(n)$. At the initial association p is bounded by n since there are atmost n wavefront segments A such that $RV(A)$ contains an edge only associated with A. At some stage in the algorithm a wavefront segment, A, may be in a situation that $RV(A)$ again does not contain any edge associated only with A. We let s be the number of segments in

$SR = \{f | f$ is only associated with a boundary edge which has more than one flow segment associated with it $\}$.

$s - 2$ of these flow segments have RV empty. Either RV was empty initially from before or it becomes empty. It becomes empty when RV is modified due to a merger of waveform segments. The number of mergers are bounded by $O(n)$. Thus the sum total of all segments in SR during the entire algorithm is $O(n)$. The bound follows.

Next consider the bound on q. Each contiguous sequence of boundary segments removed, say s_1, is charged either to an increment in SR (since an arc may now be associated only with an edge which has other associated arcs) or to the split of the contiguous sequences itself. Since changes in SR have been bounded by $O(n)$, a bound of $O(n)$ will follow on q from the total number of splits possible. The number of splits is bounded by n and the number of merges of two contiguous sequences that occur. The merge of two boundary sequences A and B follows when the association of one sequence changes to the arc associated with the other. At this step the association of an arc may be completely or partially eliminated. This is charged to either an I-intersection event, which are $O(n)$ in number since at each such event an arc is eliminated from the wavefront, or to a merge operation which again are $O(n)$ in number. This gives an $O(n)$ bound. ■

We summarize the above discussion as follows:

Lemma 2.9 *$RV(A)$, boundary sections and waveform-sections are correctly updated when the boundary changes.*

Proof: The boundary changes with the waveform striking a boundary edge which leads to another triangle. Alternatively, the boundary changes when the waveform strikes a vertex.

Consider an edge strike. If the triangle is not used by the GSP-Waveform the waveform and boundary-sections are correctly updated by association of arcs to the new boundary edges. Alternatively, if the triangle is used already by the waveform, a merge is used to update the associations. $RV(A)$ is updated correctly as shown in Lemma 2.8 and hence so are the boundary and waveform sections at this step. If the boundary changes at a vertex strike then shortest geodesic paths which pass through the vertex, say v, are required to be generated. These faces are correctly determined using the characterization in Lemma 2.2 and for each face a waveform arc generated with the vertex as center.

■

Finally, we state the result which will be useful to the overall complexity bound.

Lemma 2.10 *The number of times triangle edges are struck by the GSP-Waveform is $O(n)$*

Proof: We consider edges of the triangulated polygonal scene. There are $O(n)$ edges initially. An edge, e when struck by a waveform arc from inside a triangle T is on the boundary. After the strike it no longer is on the boundary. Any further merges of waveforms in the triangle T will not use e as part of boundary-sections since they only use edges on the current boundary to further construct boundary-sections as described in the merge procedure above. Finally, note that an edge can be used at most twice as part of the boundary, once each from the two faces it bounds. ■

2.3.4 Waveform changes

In this section we consider changes to the GSP-Waveform sections and the C-Waveform sections due to expansion of the waveform. We also show how to maintain the C-Waveform sections.

The maintenance uses red-black balanced trees. As has been described before, at the leaves are arcs in the sequence that they occur in the waveform. Internal nodes store tangents which are bridges between the upper hulls of waveforms defined by the two subtrees. Also stored at each internal node is an angle $Ang(v)$ which represents the angle of rotation of the centers of waveform arcs in one of the subtrees, say the left subtree rooted at v. This angle aligns the two unfoldings one corresponding to the arcs in the left subtree and the other corresponding to the ones in the right subtree. The rotation of the center of an arc at a leaf node, l, is obtained from the rotations along the boundary from the leaf to the root node.

We first consider changes to the waveform due to increase in the distance. The increase in distance causes the radius of arcs to increase which can lead to an arc shrinking and eventually disappearing so that its neighbors are now adjacent arcs. This involves removal of the arc from the waveform and is detected using I-curves separating the arc from its neighbors. Also two non-neighboring arcs may intersect and eliminate all arcs lying in between. And lastly two arcs in two different waveform-sections may intersect eliminating arcs in-between.

Secondly, we will consider changes to convex hulls at the nodes of the balanced tree due to increase in the distance and hence a change in the nature of the C-waveform section. These events do not change the sequence of arcs in the waveform but only affect the bridge constructions.

GSP-Waveform Changes:

We first consider the case when an arc disappears due to shrinkage of size. This is detected as follows: Let A_1, A_2, A_3 be three arcs with defining vertices c_1, c_2 and c_3. Let I_1 and I_2 be the curves defining points equidistant from the pairs (c_1, c_2) and (c_2, c_3) respectively. Then A_2 is removed when I_2 and I_1 intersect. And new events, i.e. the removal of A_1 and A_3 due to adjacent arcs is computed. The path-section associated with A_2 is removed and path-sections updated due to the partition obtained by the new I-curve. The changes in the path-sections are of two kinds, one in the path-section corresponding to A_2 itself and the other in adjacent path-sections associated with waveform segments adjacent to A_2.

To determine the path changes , given an I-curve, procedure $I - intersect$ determines the edge of the path-section P, that the I-curve strikes.

The above procedure requires $O(logn)$ steps. Using this, one can determine associations of edges in P_1 with arcs in W_n. When I-curves intersect the associations change. These intersections will be event points and the changes will form part of event point handling routines.

Since the removal of the segment as above changes the convex hulls in the balanced tree structure the capacity computations are changed. These changes are restricted along the tree path from the arc that disappears to the root and can be effected in $O(log^2 n)$ time using the balanced tree structure.

Next, consider determination of the removal of a sequence of arcs when two non-adjoining arcs meet or intersect. Since this does not affect the convex hull computations for an arc sequence associated with an edge and the shortest distance computations, this event need not be determined.

The case when two waveform-sections meet is similar to the detection of non-adjacent arc-arc intersection.

We can thus state:

Lemma 2.11 *The C-Waveform sections and C-Boundary sections are correctly maintained under changes in $O(log^2 n * (CNS + 1))$ steps.*

Convex Hull Changes

We finally consider the changes in the bridges at the nodes of the tree storing the GSP-Waveform-sections. The endpoint of a bridge is either a common tangent to a circular arc or it is the endpoint of the arc. When the bridge is a common tangent, the endpoint of the bridge may change when the endpoint of the arc becomes the endpoint of the common tangent. This happens when the normal to the bridge at its endpoint and the I-curve separating the arc from its neighbor intersect. This event changes convex hulls along the tree path to the root. These hulls can be updated in $O(log^2 n)$ steps.

Lemma 2.12 *The C-Waveform sections are correctly maintained under convex hull changes in $O(log^2 n)$ steps.*

2.3.5 Tree Changes

In this sub-section we consider maintaining the convex hull trees storing the GSP-Waveform-sections under changes due to operations which change the structure of the tree, i.e. *tree operations*. We also show how to maintain the convex hull tree storing boundary-sections under tree operations.

We consider the basic operations of splits and merges in the convex hull trees. Insertions into a set of arcs is achieved by splits and merges. Each split and merger requires $O(log^2 n)$ time to implement since convex hulls along a leaf to root path are to be constructed. Besides updates of convex hulls at each of the nodes affected by the split or merge, the rotations associated with the nodes are also updated. Since the trees are balanced the tree height remains $O(\log n)$. The effect of the tree operations on the capacity computations and other event points will be dealt with in the next section.

A dynamic maintenance strategy, similar to the one above, can be applied to maintain the C-Boundary sections. The maintenance in this case is simpler since we do not have to compute the intersections between vertices, neighboring or non-neighboring.

2.3.6 Event Point Updates

In this section we consider the maintenance of events after each event has occurred. There are two types of event points:
(1) Boundary edge strikes computed by shortest distances between waveform and boundary-sections.
(2) I-intersections, N-intersections and Convex Hull Changes in the C-Waveform sections.
We consider the two basic types of changes at the events:
(1) Boundary changes.
(2) GSP-Waveform changes.
and their effect on the event points. We also describe the effect of Tree Changes on the event points.

On the occurrence of a Boundary change or a GSP-Waveform change, the C-Waveform and C-Boundary sections are updated. The effect on the first type of event points is considered first. The shortest distances between the changed Waveform-section and the associated Boundary-section is then updated in $O(logn)$ steps for each changed section. Since changes in the C-Waveform and C-Boundary sections are implemented as splits and merger of trees, we consider later the effect of splits and merger of trees on the second type of event points.

We next consider how GSP-waveform changes, apart from those which create splits and merges of trees, affect the second type of events. The effect of an I-intersection event is to compute a new I-curve and a new I-intersection. N-intersections also generate a new I-curve and new I-intersections. Bridge splits generate new bridge splits. Moreover, Convex Hull changes also affect convex hulls along one tree path in the tree as well as bridge split events which are updated.

Given updated GSP-Waveform and boundary-sections, the time required for computation or update of the shortest distance between boundary-sections and waveform-sections is $O(logn*(1+No.of\ Bridge\ Splits)+logn*(CNS))$ where CNS is the number of boundary-sections changed. The other changes, apart from splits and merges in the trees, which we discuss below, require $O(logn)$ steps per event point.

At a tree merge or split, I-intersections and Convex Hull changes can easily be updated or computed in $O(log^2 n)$ steps along the path in the tree where a split or a merge takes place. N-intersections need not be considered explicitly as described before.

Lemma 2.13 *All event points can be determined and updated in $O(log^2 n)$ amortized operations.*

Proof: The determination of the first kind of events involves distance computations which require $O(logn)$ for each bridge split. The total number of bridge splits are $O(n)$ since there are $O(n)$ bridges in the convex hull trees. New bridges may be created since bridges may change at rotations. However the total number of bridges changed over a sequence of $O(n)$ rotations is $O(n)$. Updates to capacity computations also require the same amount of time. The second type of event points require computing I-intersections and Bridge splits and Convex Hull changes which have been shown to be determined in $O(log^2 n)$ amortized operations above. Updates of these events due to splits and merges of the tree have also been shown to require $O(log^2 n)$ time. Finally, the Convex Hull changes require $O(1)$ steps to determine and can be updated in $O(logn)$ steps at all the events. ■

2.4 Algorithm Summary

We summarize our algorithm now: At each step of the algorithm $C-Waveform$ and $C-BoundarySections$ are maintained. A heap stores the events that cause changes to these sections. Each entry of the heap defines an event that gives the distance by which the GSP-Waveform advances so as to result in a change in the waveform or the boundary. The algorithm terminates when the waveform reaches the sink vertex.

2.5 Correctness and Complexity.

To prove the correctness of the algorithm we need the following lemma, where monotone paths are paths with points in order of increasing geodesic distance from s:

Lemma 2.14 *The GSP-Waveform cycle representing the points at shortest geodesic distance d on monotone paths between s and t is correctly computed.*

Proof. The correctness follows from the fact that that the GSP-Waveform and boundary-sections are correctly maintained. Thus distance computations are correctly obtained. That waveform and boundary-sections are correctly maintained (modulo event point changes) can be proven by induction on the number of changes made to the waveform. We partition the unfoldings into sets s.t. two different sets always contain unfoldings which do not have any common face except the face with the source vertex. The initial unfoldings form disjoint sets. If unfolding U_1 merges with U_2 at a common face then the corresponding sets are merged. For a given set of unfoldings, $S(UF)$, we consider the sequences of edges which form a boundary cycle B_S which encloses the region containing the destination point. Moreover let $B(UF)$ be the bounding edges on the unfolding UF. The following invariant is maintained for the edges in $B(UF)$, which have at least one vertex in common with B_S, and the waveform W: The boundary and waveform are partitioned into sections such that each edge, e of the boundary is associated with the arc or arcs closest to it. This invariant is maintained modulo further events. There are two types of changes during the execution of the algorithm, one where the waveform alone changes (either due to an arc being eliminated by its adjacent arcs or by non-adjacent arcs) and the other when boundary strikes occur. These boundary strikes could be on the boundary cycle or on other bounding edges which lead to the boundary edges on the cycle. By Lemmas 2.11, 2.12 and 2.9, after each change the required structure of the waveform is maintained and the convex hulls are maintained to give the shortest distance between boundary and the waveform. The details of the changes have been outlined before. ■

We next consider the time complexity. We first show the following lemma:

Lemma 2.15 *The total number of event points during the course of the algorithm is $O(n)$.*

Proof. We consider the various events. First consider the boundary changes. The number of boundary changes is $O(n)$ as shown in Lemma 2.10.

We next consider the event points obtained due to GSP-waveform changes. Firstly we show that the number of arcs in the waveform sections is $O(n)$. New arcs are generated whenever faces are struck at vertices. There are thus $O(n)$ distinct circular arcs generated. There are three types of events, one where the waveform changes due to I-intersections and the other due to N-intersections and Convex Hull changes. There are only $O(n)$ I-intersections since at each intersection an arc disappears never to occur again in the GSP-Waveform. N-intersections are not computed. Finally, there are $O(n)$ Convex Hull Changes since each bridge contributes only a constant number of such changes. ■

We next bound the number of sections that can be created.

Lemma 2.16 *The number of C-Waveform sections and C-Boundary sections created is $O(n)$.*

Proof: First consider boundary-sections. We bound the number of boundary-section creations in terms of the number of arcs and the event points that occur during the algorithm. We consider boundary-sections containing more than one boundary edge since the convex hull of these only is maintained. Each boundary-section is associated with an arc. The number of such arcs is $O(n)$ as shown in the previous lemma. This boundary-section is associated with an arc, say seg, when seg is not part of a waveform-section in a triangle and the arc, seg, is charged once when the boundary-section is created. Further changes to the associations occur when for boundary-sections, arc associations are changed so that the arc is no longer associated with a boundary-section.

Firstly, this may occur at an event where the structure of the waveform changes. The boundary-section now has a different arc association. The change of association also changes the charge assigned to the arc. For I-intersections, the boundary-section change is charged to the event point that changes the association. And for N-intersections, the changes are charged to the arc eliminated.

Secondly, changes to the set of boundary-sections take place when the boundary changes. These are charged to the boundary changes. This happens at a merge step when multiple arcs at an edge are associated with other edges. These arcs have not been associated with boundary-sections before and the creation of the boundary-sections is charged to these arcs.

Finally, note that the waveform-sections are only $O(n)$ since there are only $O(n)$ vertices and edges that give rise to waveform arcs. ■

We obtain the following theorem:

Theorem 2.1 *The geodesic shortest path problem can be solved in $O(nlog^2n)$ operations.*

Proof: To prove the final time complexity bound note that the algorithm processes data at event points where each event point determination requires $O(log^2n)$ amortized operations (Lemma 2.7). A time bound of $O(log^2n)$ is required to update event points. There are $O(n)$ event points and $O(n)$ updates thus giving a time bound of $O(nlog^2n)$ steps.

The processing of event points requires maintaining boundary and waveform-sections. The processing of the boundary and waveform-sections require maintaining the sections under
(i) Boundary changes which require $O(nlogn + nU)$ steps (by Lemma 2.8) where U is the time required to update $CRV(A)$ which is $O(log^2n)$.
(ii) Waveform changes which require $O(log^2n * (CNS + 1))$ where $CNS =$ No. of sections changed which is $O(nlog^2n)$ operations combining Lemma 2.6 , Lemma 2.11 , and Lemma 2.16 and the fact that the number of sections removed is bounded by the number created. ■

References

[1] [HS] *Efficiently computing Shortest Paths*, J. Hershberger and S. Suri, Proceedings of FOCS, 1993.

[2] [KM1] *Efficient Algorithms for Euclidean shortest paths and visibility problems amongst polygonal obstacles*, Kapoor Sanjiv and S.N. Maheshwari, Proc. ACM Symposium on Computational Geometry, 1988.

[3] [KM2] *An efficient algorithm for the Geometric Max Flow and Shortest Path Problem*, S. Kapoor and S.N. Maheshwari, Manuscript, March 1991.

[4] [LP] *Euclidean Shortest Paths in the Presence of Rectilinear Barriers*, D.T. Lee and F.P. Preparata., Networks, vol.14 (1984), pp.393-41O.

[5] [Mi] *Planning Shortest Paths*, J.S. Mitchell, Ph.D dissertation, Dept. of Operations Research, Stanford University, 1986.

[6] [Mi3] *Efficiently computing Shortest Paths*, J.S.B. Mitchell, Proceedings of the 9th A.C.M Conference on Computational Geometry, 1993, pp. 341-352.

[7] [Mo] *On finding shortest paths on convex polyhedra*, D. Mount, Technical Report 1495, Dept. of Computer Science, Univ. of Maryland, Baltimore, 1985.

[8] [MMP] *The Discrete Geodesic Problem*, J.S.B. Mitchell, D.M. Mount and C.H. Papdimitriou, SICOMP Vol. 16, No. 4. August 1987.

[9] [OL] M. Overmars and J. Van Leeuwan, Maintainence of configurations in the plane, Journal of Computer and System Sciences, 23, pp.166-204, 1981.

[10] [P] *Shortest Path Motion*, C.H. Papadimitrou, Proceedings 6^{th} FST & TCS, New Delhi, India, LNCS 241, pp.144-153.

[11] [P2] *An Algorithm for shortest-path motion in three dimensions*, C.H. Papadimitrou, IPL, 20 (1985), pp. 259-263.

[12] [RS] *Shortest Paths in Euclidean Space with Polyhedral Obstacles*, J.Reif and J. Storer, JACM 1992.

[13] [RSB] *Shortest Paths on Polyhedral Surfaces*, J. O'Rourke, S.Suri and H. Booth, Manuscript, Johns Hopkins Univ., Baltimore, MD, 1984.

[14] [SS] *On shortest Paths in Polyhedral spaces*, M. Sharir and A. Schorr, SICOMP, 15 (1986), 193-215.

3 Appendix

3.1 Shortest Distance Computations

Case 1: The first kind of distance computations requires computing the minimum distance between a waveform-section and an edge or vertex of a triangle. These distances are computed from the balanced tree structure storing the convex hulls.

The procedure starts at the root. At a node v of $CWT(W_s)$ either the shortest distance is to the current bridge or to a arc or bridge in the subtree rooted at v. Note that at each node is stored an angle which represents the change in orientation required to merge the waveforms. The procedure uses I-curves at the bridges to branch either left or right at a node, taking into account the orientation changes required at each node, until a candidate arc is found. Note that it is possible that no decision can be made on the branching. This happens when the two I-curves at a bridge intersect at a distance less than the distance from the vertex or edge to the bridge to which the shortest distance is being computed. But this involves a change in the waveform which forms an event that will be discused in the next section. Distance computations will be updated at this event

Case 2: The second kind of distance computations requires computing the shortest distance between an arc and a sequence of vertices and edges of the boundary. This distance is obtained from $CPT(P)$, the balanced tree storing the boundary-section. The shortest distance computation between an arc and a boundary-section proceeds as follows: Start at the root node of the balanced tree, CPT. At any node $v, v \in CPT(P)$, if the arc does not intersect the current bridge, taking into account the orientation stored at the node, then the shortest distance either is to the bridge or lies to its left or right. In the second and third sub-case a branch is made to the left or right and the shortest distance found recursively. On the other hand if the arc either intersects the bridge or the shortest distance found is to the bridge then let C_1 and C_2 be the convex hulls on the left and right which are removed by the bridge construction. This event is a split of the convex hull to generate two hulls and is called a *Bridge Split*. The shortest distance to these hulls is computed recursively. The shortest distance is the minimum of the two shortest distances found recursively.

We thus obtain the following result:

Lemma 3.1 *Given a $C-Waveform$ Section and a $C-$Boundary Section, the distance to the boundary-section can be computed correctly in $O(logn * (No.\ of\ Bridge\ splits))$ operations.*

Proof: We consider the cases described above separately: The case of the distance between a vertex or an edge to a sequence of arcs is simple and requires one search down the balanced search tree structure. The case of the distance between an arc and a boundary-section requires splits in the convex hull tree storing the boundary-section. For each split two $O(logn)$ searches are required to find the next bridge on each convex hull closest to the given arc. ■

Finally, we note that the assumption that no change takes place in the GSP-waveform is removed when we consider events that change the structure of the waveform. At these events the distance computations affected will be updated.