

This document contains the draft version of the following paper:

S. Dhaliwal, S.K. Gupta, J. Huang, and A. Priyadarshi. Algorithms for computing global accessibility cones. *ASME Journal of Computing and Information Science in Engineering*, 3(3):200-209, 2003.

Readers are encouraged to get the official version from the journal's web site or by contacting Dr. S.K. Gupta ([skgupta@umd.edu](mailto:skgupta@umd.edu)).

# Algorithms for Computing Global Accessibility Cones

Savinder Dhaliwal, Satyandra K. Gupta<sup>1</sup>, Jun Huang, Alok Priyadarshi

University of Maryland

College Park, MD 20742

## ABSTRACT

This paper describes algorithms for computing global accessibility cones for each face (i.e., the set of directions from which faces are accessible) on a polyhedral object. We describe exact mathematical conditions and the associated algorithm for determining the set of directions from which a planar face with triangular boundary is inaccessible due to another face on the object. By utilizing the algorithm to compute the exact inaccessibility region for a face, we present algorithms for computing global accessibility cones for each face on the object. These global accessibility cones are represented as a matrix structure and can be used to support a wide variety of accessibility queries for the object.

## 1 INTRODUCTION

Accessibility analysis of an object helps in process planning in a number of different manufacturing applications. For example, accessibility analysis is required in

- *Machining*: A good setup planning ensures high productivity as well as machinability. Accessibility analysis helps in determining machinability by finding the set of directions from which the part may be approached by the cutting tool. It also helps in determining the work-piece orientations that would minimize the number of set-ups required for machining the part [1][2] and in cutter path planning [3].
- *Assembly*: It helps in determining the directions from which the assembly and disassembly operations can be carried out.
- *Mold Design*: The mold assembly needs to be disassembled to eject the molded part. Accessibility analysis is used in accessibility/disassembly-based decomposition of the gross mold to ensure part ejection. It helps in selecting the parting surface that minimizes or eliminates the undercuts. It also helps in reducing the number of side cores required in the mold design [4][5][6].
- *Inspection Planning*: It is used in automatic planning and programming tasks with a Coordinate Measuring Machine (CMM). It helps in determining part orientation on the CMM and identifying the directions from which a probe can approach the part to perform measurements [7].

This paper describes algorithms for computing global accessibility cones for polyhedral objects modeled using facets (planar faces with triangular boundaries). If an object has curved faces,

---

<sup>1</sup> Corresponding Author.

such faces are faceted and approximated by small triangles. If the object has non-triangular planar faces, then such faces are triangulated. This paper also provides several examples to show computational performance of the algorithms.

## 2 RELATED WORK

A *Gaussian map* of a surface is the set of end points of the unit normal vectors of the surface. Gaussian maps can be represented as a spherical region (i.e., a subset of the boundary of a unit sphere). By extending the basic idea behind Gaussian maps, Woo *et al.* developed the concept of visibility map to represent and compute accessibility [4][8]. A visibility map is a set of points on a spherically convex region. Any point in a visibility map denotes a direction from which the entire surface is accessible to its exterior. Local accessibility of a point on a surface is defined by the hemispherical region constructed by using the surface normal at the point as the pole (for detailed definition of poles and hemispherical regions, please see Section 3). Therefore, the visibility map of a point is a hemispherical region on a unit sphere. The visibility map of a surface is the intersection of visibility maps of all the points on the surface. For example, the visibility map of a planar surface is a hemisphere. The concept of visibility maps has been extended by Kim *et al.* to cover bezier surfaces [9]. They have defined and provided algorithms for computing tangent, normal and visibility maps for regular bezier surfaces. Elber *et al.* [10] presented an approach to compute “visibility set” (a very similar concept to visibility map) for freeform surfaces.

The boundary of polyhedral parts can be divided into convex (portion of boundary that is part of convex hull) and concave (portion of boundary that is not a part of convex hull) regions. The visibility map of each face within the convex region is the hemisphere formed by using the direction normal of the facet as a pole. The visibility map of each concave region is the intersection of the visibility maps of all the planar faces within the region. Sometimes visibility maps cannot be used to determine the global accessibility of an individual facet within a concave region. This is because the visibility map of an object is constructed using the local accessibility information for various facets.

Suh and Kang developed an approach for performing accessibility analysis for NC machining [2]. They compute accessibility by constructing the binary spherical maps. The part surface is faceted into triangular patches. The unit hemisphere is also faceted using spherical triangles. Accessibility is computed by projecting centroids of various facets on the unit sphere and identifying the spherical triangles that contain them. Due to approximations, this approach is prone to the following two types of errors. First, it might report that an entire facet is accessible in a certain spherical triangle while actually only a portion of the facet is accessible. Second, it might report that a facet is not accessible from an entire spherical triangle while actually the facet is accessible from a portion of the spherical triangle.

Recently, methods have been developed to perform accessibility analysis by taking advantage of computer graphics hardware [3][7]. Graphics cards make use of the depth-buffer implemented using hardware to perform fast hidden surface removal and render the object in a given scene. If all the individual faces on the object have been assigned different colors, then the accessibility of each face in a given direction can be detected by rendering the object using the given direction as the viewing direction, and querying the colors that appear on the pixel map after rendering. Since each rendering actually corresponds to a particular viewing direction, the point accessibility can be approximated by sampling a finite number of directions on the Gaussian Sphere. This

approach involves two types of approximations. First, it uses finite sampling of viewing directions on Gaussian sphere. Second, it assumes that the face is so small that presence of a single pixel on the rendered scene can identify its accessibility. Therefore, the results produced by this approach are only an approximation of the exact solution.

Spitz *et al* have also done work in the area of accessibility analysis using graphics hardware [7]. They use cubic maps to approximate spherical maps and use graphics hardware to determine accessibility.

Stage and Roberts described a framework for representing and computing tool accessibility from manufacturability evaluation point-of-view [11]. This is primarily a feature-based approach, focusing on the shape/size compatibility between a tool and an entity (a face or a set of faces) to be machined. The advantage of this approach is that it works for an object with curved surfaces without any need for faceting. However, the notion of accessibility is closely tied with a particular tool.

### 3 MATHEMATICAL DEFINITIONS

In this section we present the definitions and basic mathematical concepts that are needed to describe our algorithms.

#### 3.1 Accessibility Definitions

- *Accessibility of a point in a given direction*: A point belonging to a geometric entity is *accessible* in a given direction if a ray of can be drawn from it in the given direction without intersecting with the interior of the geometric entity.
- *Accessibility of a face in a given direction*: A face is *accessible* in a given direction if all points in the interior of the face are accessible in the given direction. The boundary points of the face are not considered in this definition in determining the accessibility of a face.
- *Global accessibility cone of a point*: Global accessibility cone of a point represents the set of unit vectors along which the point is accessible.
- *Global accessibility cone of a face*: Global accessibility cone of a face represents the set of unit vectors along which the face is accessible.

#### 3.2 Review of Spherical Geometry Properties

To make this paper self-contained, we describe the following properties and definitions from spherical geometry area [12]:

*Property 1*: The curve of intersection of a plane and a sphere is a circle. It is called a great circle if the plane passes through the center of the sphere, and a small circle if the plane does not pass through the center of the sphere. The radius of a great circle is equal to the radius of the sphere.

*Property 2*: Only one great circle can be drawn through two points of a sphere that are not diametrically opposite.

*Property 3*: Two great circles intersect at two points that are diametrically opposite.

*Definition 1*: A *spherical polygon* is a portion of the surface of a unit sphere that is bounded by the arcs of great circles.

*Definition 2:* The position vector  $\mathbf{S}_p$  of the *spherical projection* of a point  $P$  onto a unit sphere is a unit vector parallel to the line that joins the center  $O$  of the unit sphere and the point  $P$ .  $\mathbf{S}_p$  is given by

$$\mathbf{S}_p = \frac{\mathbf{P} - \mathbf{O}}{|\mathbf{P} - \mathbf{O}|}$$

where  $\mathbf{P}$  is a position vector that corresponds to the point  $P$  and  $\mathbf{O}$  is a position vector that corresponds to the center of the sphere  $O$ .

*Definition 3:* The *spherical projection*  $f_p$  of a face  $f$  is the spherical projection of all the constituent points of the face. Spherical projection of a point that coincides with the center of the sphere is not defined.

*Definition 4:* A *hemisphere* is a portion of a spherical surface obtained by splitting the sphere surface by using a plane passing through the center of the sphere. The normal vector of the plane pointing towards the hemisphere is called the *pole* of the hemisphere.

### 3.3 Review of Polyhedral Part Properties

A faceted object boundary consists of two types of facets: convex-hull facets and non-convex hull-facets. *Convex-hull facets* are those facets on the part that are also on the convex hull of the part. All the facets on the part other than convex-hull facets are called *non-convex-hull* facets. Connected sets of non-convex-hull facets form *concave regions*.

*Property 4:* The global accessibility cone of a convex-hull facet is a hemisphere generated using the direction normal of the facet as its pole [4].

*Property 5:* A non-convex-hull facet can be blocked only by a non-convex-hull facet present in the same concave region [4]. A ray emanating from a point in a concave region will either intersect a facet in the same concave region or go to infinity.

*Property 6:* A pair of facets can obstruct their mutual accessibility only if they are facing each other. Since the boundary of the object is continuous, if the facets are not facing each other, there will always be at least one facet in between the two facets.

### 3.4 Problem Formulation

The input to the accessibility analysis algorithms is a polyhedral object modeled using facets (triangular planar faces). If an object has curved faces, such faces are faceted and approximated by small triangles. If the object has non-triangular planar faces, such faces are triangulated.

The output of the algorithms is the global accessibility cones of all facets on the object. The global accessibility cones are represented as a matrix called *AccessStatus* matrix. In this scheme, the boundary of the unit sphere is partitioned into a finite number of spherical triangles, each representing a set of directions. Rows of this matrix represent spherical triangles. Columns of this matrix represent the facets. Each entry in the matrix describes whether a facet is completely accessible from a spherical triangle or not. For example if  $j^{th}$  facet is completely accessible from  $i^{th}$  spherical triangle, entry that corresponds to  $i^{th}$  row and  $j^{th}$  column is set to TRUE. On the other hand, if  $j^{th}$  facet is not accessible from  $i^{th}$  spherical triangle, the entry corresponding to the  $i^{th}$  row and  $j^{th}$  column is determined in the following manner. If the  $i^{th}$  spherical triangle does not reside within the visibility map of the  $j^{th}$  facet (a unit hemisphere with the normal vector of the  $j^{th}$  facet as the pole), the entry is simply set to FALSE. Otherwise, if  $i^{th}$  spherical triangle does reside

within the visibility map of  $j^{\text{th}}$  facet, the entry is filled with the index of the facet(s) that is blocking  $j^{\text{th}}$  facet from the directions represented by  $i^{\text{th}}$  spherical triangle.

The status of the boundary of each spherical triangle is defined as follows. Since we consider the global accessibility cones as close sets, if a facet is completely accessible from directions in a spherical triangle, then it is also completely accessible from directions on the edges of this spherical triangle.

Since the global accessibility cone of a convex-hull facet is always a hemisphere (property 4), we only need to compute the global accessibility cones for non-convex-hull facets. Therefore, only the accessibility information of non-convex-hull facets is needed to be stored in the *AccessStatus* matrix.

## 4 COMPUTING EXACT INACCESSIBILITY REGION OF A FACET DUE TO PRESENCE OF ANOTHER FACET

In order to compute the global accessibility cone for a face on a polyhedral object, we will first investigate how presence of a facet makes some other facet inaccessible from certain directions. Given two planar facets  $f$  and  $f'$ , we are interested in determining the set of directions from which  $f$  becomes inaccessible due to presence of  $f'$ .

### 4.1 Mathematical Definition of Inaccessibility Region of a Facet due to Presence of another Facet

Let  $f$  and  $f'$  be two facets on the boundary of an object. Let  $I$  be the set of directions (i.e., unit vectors) from which  $f$  is *inaccessible*. Let  $V$  be the set of directions (i.e., unit vectors) from which  $f$  is *accessible*. We also refer to  $I$  as *inaccessibility region* and  $V$  as *accessibility region*.  $I$  and  $V$  can be constructed in the following manner:

- 1) Set  $I = \emptyset$ ,  $V = H$ , where  $H$  is a unit hemisphere defined using outward normal vector of  $f$  as the pole.
- 2) For every point  $p$  in the interior of  $f$ , do the following:
  - Construct a unit sphere  $S$  centered at  $p$
  - Compute the spherical projection  $f_p'$  of  $f'$  on  $S$
  - $I = I \cup (\text{interior of } f_p')$
- 3)  $V = V - I$ .

Though this is a mathematically rigorous construction of accessibility and inaccessibility regions, it is computationally impractical because there would be infinite number of points on the facet  $f$  for which the projection of  $f'$  would need to be computed. Therefore, we need to develop a computationally practical but mathematically equivalent algorithm. Section 4.2 describes such an algorithm. Before defining the algorithm, we need to present Proposition 1.

**Proposition 1:** Inaccessibility region  $I$  for a facet  $f$  due to another facet  $f'$  is a convex region.

**Proof:** Let  $A$  be the spherical triangle formed by the projection of a point in facet  $f'$  (for example the centroid of  $f'$ ) onto the sphere as the sphere moves over facet  $f$ . Let  $B$  be the interior of a spherical triangle representing the spherical projection of  $f'$ . Inaccessibility region  $I$  is formed by sweeping  $B$  over  $A$ . Both  $A$  and  $B$  are convex regions. Therefore,  $I$  is also a convex region.

## 4.2 Algorithm for Computing Exact Inaccessibility Region

Given two planar facets  $f$  and  $f'$ , the set of directions from which  $f$  is inaccessible due to  $f'$  can be computed using the algorithm defined below. Once again, let the set of directions from which  $f$  is inaccessible due to  $f'$  be given by  $I$  and the set of directions from which  $f$  is accessible be given by  $V$ .

**Algorithm** INACCESSIBLE( $f, f'$ )

- 1) Construct unit spheres at the three vertices of the facet  $f$ . Project  $f'$  on these spheres and let the projections be denoted by  $f'_{P_1}, f'_{P_2}$  and  $f'_{P_3}$ . If any vertex of  $f'$  lies at the center of sphere, then do not include that vertex in the projection.
- 2)  $f'_{P_1}, f'_{P_2}$  and  $f'_{P_3}$  are spherical triangles (or arc if the two facets share vertices) on the sphere and would result in maximum of nine vertices on the unit sphere.
- 3) Find the convex hull  $C_V$  of the vertices computed in the previous step.
- 4)  $I =$  the interior of  $C_V$ . ( $I$  is a convex spherical polygon representing the set of directions from which  $f$  is inaccessible due to  $f'$ .)

In the next section we will prove that the algorithm described in this section is mathematically equivalent to the definition of inaccessibility region defined in Section 4.1.

## 4.3 Correctness Proof for Inaccessibility Region Determination Algorithm

In this section we prove that the algorithm described in the Section 4.2 for computing  $I$  is equivalent to the mathematical definition of  $I$  given in Section 4.1. To do this, we will use one of the fundamental properties of convex regions that states that the convex hull of a convex region is the convex region itself. Using this property of convex region, we will prove that the convex hull generated by the algorithm described in Section 4.2 is equivalent to the convex inaccessibility region defined in Section 4.1.

As a preparation for the main proof, two basic spherical projection characteristics will be proved as Proposition 2 and 3. With these two basic characteristics, the whole proof proceeds as follows. We will first prove that the boundary of inaccessibility region defined in Section 4.1 is a convex spherical polygon. This will be done in two steps. In the first step we will prove that the boundary of the inaccessibility region can be produced by taking the projection of facet  $f'$  onto the unit sphere by moving the sphere only over the edges of facet  $f$ . Proposition 4 will provide this portion of the proof. We will then prove that as the unit sphere moves over edges of facet  $f$ , the projection of facet  $f'$  would result in a convex spherical polygon. Propositions 5 and 6 will provide this portion of the proof.

Once we have proved that the boundary of inaccessibility region is a convex spherical polygon, we just need to know the vertices of this spherical polygon in order to generate the convex hull and hence the region itself. We will show that the points obtained by projecting facet  $f'$  onto unit spheres centered at the three vertices of facet  $f$  are the superset of the vertices that form the boundary of the inaccessibility region. We will prove this by showing that three arcs generated by projecting the vertices of facet  $f'$  onto the sphere as it moves along an edge of facet  $f$  do not intersect. Proposition 7 described below provides this portion of the proof. Therefore no new vertices are created as the facet  $f'$  is swept over an edge of facet  $f$ . Hence the points obtained by projecting facet  $f'$  on unit spheres centered at the three vertices of facet  $f$  indeed are the superset of the vertices that form the boundary of the inaccessibility region.

**Proposition 2:** Spherical projection of a line segment not passing through the center of the sphere is a great arc (This proposition is a minor modification of the proposition described in [13]).

**Proof:** The spherical projection of a line segment on a unit sphere is the set of spherical projection of all the points constituting the line segment. We know from plane geometry that 3 non-collinear points can be used to define a unique plane. Therefore if the center of the sphere is not collinear with the vertices of the line segment ( $PQ$ ), the vertices of the line segment along with the center of the unit sphere ( $O$ ) define a unique plane. From Property 1, the intersection of this plane with the sphere would generate a great circle. Since the line segment lies on the plane, its projection would be a portion of this great circle, which is a great arc.

**Proposition 3:** Spherical projection of a point onto a unit sphere moving along a line segment  $AB$  is an arc of a great circle. This projection is equivalent to the projection of line segment  $PQ$  onto a unit sphere centered at point  $A$ , where  $PQ$  is a line segment of same length as  $AB$ , and parallel, but apposite in direction to  $AB$ .

**Proof:** Let the point being projected onto the unit sphere be denoted by  $P$ . We are interested in finding the locus of the projection of  $P$  onto the unit sphere (initially located at  $A$ ) as the sphere moves from  $A$  to  $B$  along the line segment  $AB$ . From rigid body transformation, we know that the transformation of the sphere relative to the point  $P$  is equivalent to the inverse transformation of the point  $P$  relative to the sphere. Since in this case the transformation of the sphere is a linear translation along line segment  $AB$ , its inverse transformation for  $P$  would also be a linear translation in a direction apposite and parallel to  $AB$  and also the same length as  $AB$ . Let us assume that this transformation would result in a line segment  $PQ$ . Thus, the locus of the projection of the point  $P$  onto a sphere, initially centered at  $A$ , as it moves along the line segment  $AB$  is equivalent to the projection of the line segment  $PQ$  onto the sphere located at  $A$ . From Proposition 2, we know that this projection would be an arc of a great circle.

**Proposition 4:** Inaccessibility region  $I$  of a facet  $f$  can be determined by the inaccessibility region of its edges.

**Proof:** Consider the projection of a point  $P$  onto a unit sphere as it moves over the edges of a facet  $f$ . Following the argument presented in Proposition 3, instead of moving the sphere, we can move the point along a line segment. Therefore, the projection of the point  $P$  onto a sphere as it moves over the edges of a facet  $f$  is equivalent to the projection of a triangle  $t$ , having the same dimensions as the facet  $f$ , onto the sphere. When the sphere is centered at any point  $R$  lying inside the facet, the projection of the point  $P$  is equivalent to the projection of a point  $R'$  lying inside triangle  $t$  following the transformation argument presented in Proposition 3. The projection of triangle  $t$  onto the sphere would result in a spherical triangle. From convexity principles, since  $R'$  lies in the interior of a convex polygon (triangle  $t$ ), its projection would also lie in the interior of the spherical polygon. Hence, it is sufficient to project the edges of the triangle  $t$ , which implies that it is sufficient to examine the edges of facet  $f$ . Thus, we have proved that for finding the projection of a point onto a sphere as the sphere moves over a facet, it is sufficient to move the sphere along the edges of the facet.

Now consider the projection of facet  $f'$  on facet  $f$ . Since  $f'$  is nothing but a collection of points, therefore, by induction it holds that it is sufficient to examine the edges of facet  $f$  for finding the projection of the facet  $f'$  onto the sphere.

**Proposition 5:** The projection of a triangular planar facet onto a unit sphere as the sphere moves along a line segment is a spherical polygon.

**Proof:** From basic spherical geometry, we know that the spherical projection of a triangle is a spherical triangle. If the sphere were now moved along a line segment, the projection of the triangle on the sphere would also move. We are interested in finding the swept region that represents union of projections as the sphere moves along a line segment. From Proposition 3, we know that the spherical projection of a point on a sphere as the sphere moves along a straight line is a great arc on the unit sphere. Since a triangle can be considered as a collection of points, its projection on the sphere as the sphere moves along a straight line would be collection of great arcs on the unit sphere. Moreover, since we are considering the projection of a convex polygon on a sphere as it moves along a line, only the vertices of the convex polygon need to be projected in order to construct the boundary of the projected convex region. The three vertices would generate three great arcs as the sphere moves along the line segment. These three arcs along with the two spherical triangles (that result due to projection of the facet when the sphere is at the vertices of the line segment) are the only possible choices for forming the boundary of the projected convex region. Since only arcs form the boundary of this projected region, this region is a spherical polygon.

**Proposition 6:** Boundary of inaccessibility region  $I$  for a facet  $f$  due to another facet  $f'$  is a convex spherical polygon.

**Proof:** The closure of inaccessibility region  $I$  for a facet  $f$  due to another facet  $f'$  is formed by spherical projection of facet  $f'$  onto a unit sphere as it moves over facet  $f$ . We have already shown in Proposition 4 that we only need to consider the edges of a facet  $f$  to compute the projection of  $f'$ . We have also shown in Proposition 1 that  $I$  is a convex region. Therefore the only possible candidates for forming the boundary of  $I$  are boundaries of the three spherical polygons generated as the sphere moves along the three edges of  $f$  (as a consequence of Proposition 5). Therefore, the boundary of  $I$  will consist of only arcs and it will be a convex spherical polygon. The basic idea behind this proposition is shown graphically in Figure 1 using planar representation.

**Proposition 7:** Vertices obtained by projecting the facet  $f'$  onto the sphere centered at the three vertices of facet  $f$  form the super set of the vertices that constitute the convex hull of the inaccessibility region  $I$ .

**Proof:** To prove this we have to show that during projection of a facet  $f'$  as the sphere moves along the three edges of  $f$ , no new vertices are created (i.e., all vertices generated during this operation are included in the set of vertices obtained by projecting the facet  $f'$  on the sphere centered at the three vertices of facet  $f$ ). This implies that the three arcs that define the swept projection of  $f'$  as the sphere moves along each of the edges of  $f$  should not intersect and therefore do not create any new vertex.

Consider a line segment  $LM$  and a unit sphere that moves along a line segment  $AB$ , as shown in Figure 2. From Proposition 3 we know that for the line segment  $LM$ , the locus of projection on the sphere as it moves along  $AB$  is equivalent to the projection of a parallelogram onto the unit sphere centered at  $A$ . The lengths of the adjacent sides of the parallelogram are  $AB$  and  $LM$ . Let the four vertices of the parallelogram be given by  $L$ ,  $M$ ,  $S$ , and  $T$ . Where  $MS$  is parallel to  $LT$ . We are interested in finding the projection of  $MS$  and  $LT$  on the unit sphere.

The points  $A, B, L, T$  and  $A, B, M, S$  also form parallelograms since  $AB = LT = SM$  and  $AB \parallel LT \parallel MS$  (as a consequence of Proposition 3). From plane geometry we know that a unique plane passes through a parallelogram. Therefore parallelograms  $ABLT$  and  $ABMS$  lie on unique planes. Let these planes be given by  $P_{LT}$  and  $P_{MS}$  respectively. Since  $AB$  lies on both the planes, this implies that the line containing  $AB$  is the line of intersection of planes  $P_{LT}$  and  $P_{MS}$ . The planes  $P_{LT}$  and  $P_{MS}$  intersect the unit sphere to form two great circles. Since line segments  $LT$  and  $MS$  lie on planes  $P_{LT}$  and  $P_{MS}$  respectively, their projections on the sphere would lie on the corresponding great circles. The line  $AB$  would be collinear with the common diameter of these great circles since it lies on both planes and also passes through the center  $A$  of the sphere. From Property 3 we know that two great circles intersect at exactly two points that are diametrically opposite. Therefore, these two great circles will not intersect at any other point. Hence, the projection of line segments  $LT$  and  $MS$  will not intersect as long as points  $L$  and  $M$  are not collinear with the line segment  $AB$ . Therefore, no new vertex will be created. If  $L$  and  $M$  are collinear with the line segment  $AB$ , the projection of  $LT$  and  $MS$  would be a point on the unit sphere and therefore, no new vertex will be created.

A triangle would consist of three line segments  $LM, LN,$  and  $MN$ . These line segments would form three parallelograms. The projections of the parallel sides of these parallelograms would result in non-intersecting arcs of great circles. Therefore we will get three arcs that would be non-intersecting.

## 5 GENERATING EXACT GLOBAL ACCESSIBILITY CONES

This section describes an algorithm that generates exact accessibility cones for each facet on an object and stores the accessibility information in the matrix *AccessStatus*.

### 5.1 Overview of Approach

This approach follows an “initialize-update” scheme. It initially assumes a non-convex-hull facet  $f$  is accessible from all the orientations described by the hemisphere  $H$  created by the facet’s direction normal as its pole. Algorithm INITIALIZE described in Section 5.2 performs this initialization for all non-convex-hull facets. Number of spherical triangles on the boundary of the unit sphere (equal to number of rows in the matrix *AccessStatus*) is dynamic in nature and grows on as-needed basis. Additional rows are added in this matrix as and when finer spherical triangles are required to exactly represent global accessibility cones.

Algorithm UPDATE described in Section 5.3 updates accessibility cone of each facet by adaptively incorporating the influence of other facets. Algorithm UPDATE calls algorithm INACCESSIBLE to find the inaccessibility region of various facets. The boundary of the unit sphere is further reclassified and subdivided according to the accessibility of various facets as reported by algorithm INACCESSIBLE.

Section 5.4 discusses the implementation issues associated with this approach.

### 5.2 Initializing Accessibility Matrix to Represent Global Accessibility Cones

Algorithm INITIALIZE takes the set of non-convex-hull facets  $F_n$  of the object as an argument. It uniquely classifies the boundary of the unit sphere into spherical polygons according to the initial accessibility status of each facet. This initial accessibility information is stored in matrix *AccessStatus* and will be later updated by the iterative algorithm UPDATE.

**Algorithm INITIALIZE ( $F_n$ )**

- 1) Form the set of spherical polygons  $X$  by taking the intersection of all great circles corresponding to normal vectors of various facets in  $F_n$ .
- 2) Triangulate every polygon in  $X$ . Let  $S$  be the set of resulting spherical triangles.
- 3) Create an  $M \times N$  matrix *AccessStatus*, where  $N$  is the number of the facets in  $F_n$ ,  $M$  is the number of spherical triangles in  $S$ . Each entry of the matrix will indicate the accessibility status (TRUE or FALSE) of a facet  $f$  from a set of directions defined by a spherical triangle  $s$ . Initially, for a facet  $f$ , if a triangle  $s$  is contained by hemisphere  $H(f)$ , the corresponding entry in the matrix is set to TRUE; otherwise, the corresponding entry is set to FALSE.

**5.3 Updating Initial Accessibility Matrix**

As stated in property 5 and 6, inaccessibility region of a non-convex-hull facet  $f$ , can be determined by only considering the influence of other non-convex-hull facets facing  $f$  and present in the same concave region. Hence, to determine the accessibility cone of a non-convex-hull facet  $f$ , it is sufficient to only examine the influence of other non-convex-hull facets facing  $f$  and present in the same concave region.

UPDATE takes two arguments,  $S$  and  $F_n$ .  $S$  is the initial set of spherical triangles on the boundary of the unit sphere generated by INITIALIZE.  $F_n$  is the set of non-convex-hull facets on the object. For each pair of facets ( $f, f'$ ) in  $F_n$  that face each other and are present in the same concave region, UPDATE calls algorithm INACCESSIBLE to find the inaccessibility region of  $f$  on the spherical map due to  $f'$ . We convert the inaccessibility region  $I$  into a set of spherical triangles by triangulating it. Then, UPDATE calls PARTITIONTRIANGLES, which is an algorithm that reclassifies the spherical boundary through further partitioning, and updates the *AccessStatus*. In general, the number of the rows in the matrix *AccessStatus* grows with each partitioning.

**Algorithm UPDATE( $S, F_n$ )**

For every pair of facets ( $f, f'$ ) in  $F_n$  that face each other and are present in the same concave region, do the following:

- 1) Call INACCESSIBLE ( $f, f'$ ), which returns  $I$ , the convex spherical polygon representing the inaccessibility region of  $f$  due to  $f'$ .
- 2) Triangulate the convex polygon  $I$  into a set of spherical triangles.
- 3) Call PARTITIONTRIANGLES ( $S, I, f$ ), which returns the updated uniquely classification  $S$  of the boundary of the unit sphere.

If every triangle in set  $I$  does not exactly match with (i.e., is equal to in the set theoretic sense) some triangle in set  $S$ , then PARTITIONTRIANGLES is used to subdivide triangles in  $S$  and  $I$  to make sure that the subdivided triangles match. Every pair of spherical triangles ( $s, \Delta$ ) is examined to find out if we need to subdivide triangles (where  $s$  is an element of set  $S$  and  $\Delta$  is an element of set  $I$ ). If  $s$  and  $\Delta$  intersect, then we triangulate the intersection region using algorithm TRIANGULATE. For computing Boolean operations on spherical triangles, we first convert spherical triangles to planar triangles using central projection [12]. After this step we apply standard planar polygon intersection algorithms. TRIANGULATE takes a spherical polygon  $P$  and partitions the given polygon  $P$  into triangles and returns the result. This algorithm has been adopted from a standard triangulation procedure [14]. After subdivision of spherical triangles, we

update *AccessStatus* to incorporate information in inaccessibility region  $I$  for facet  $f$ . Based on the result of the triangulation, PARTITIONTRIANGLES updates *AccessStatus*.

**Algorithm** PARTITIONTRIANGLES ( $S, I, f$ )

For every pair of intersecting spherical triangles ( $s, \Delta$ ), perform the following steps (where  $s$  is an element of set  $S$  and  $\Delta$  is an element of set  $I$ ):

- 1) Calculate  $Int$ , the intersection of  $s$  and  $\Delta$ .
- 2) If  $\Delta$  is completely contained by  $s$ , do the following:
  - a. Let  $T_1 = \Delta$ .
  - b.  $T_2 = \text{Triangulate}(s - Int)$ .

Otherwise, do the following:

  - a.  $T_1 = \text{TRIANGULATE}(Int)$ .
  - b.  $T_2 = \text{TRIANGULATE}(s - Int)$ .
- 3) Insert rows in Matrix *AccessStatus* for all the spherical triangles in  $T_1$  and  $T_2$ .
- 4) For every spherical triangle  $t_1$  in  $T_1$ , set the value of the corresponding entry  $(t_1, f)$  in *AccessStatus* to be FALSE.
- 5) For every spherical triangle  $t_2$  in  $T_2$ , set the value of the corresponding entry  $(t_2, f)$  in *AccessStatus* to be the same as the value of entry *AccessStatus* ( $s, f$ ).
- 6) Delete row  $s$  from *AccessStatus*.
- 7)  $S = (S - s) \cup T_1 \cup T_2$ .

#### 5.4 Implementation Issues

Section 5.2 and Section 5.3 have shown that theoretically the approach described above leads to exact accessibility cones. However, numerical accuracy problems common to geometric algorithms may creep in to make the implementation difficult. We found during the implementation, for example, that the adaptive refinement may generate spherical triangles having two of the vertices extremely close to each other. This tends to cause wrong vertex classifications due to numerical inaccuracies. There has been a great deal of progress in developing techniques for robust geometric computations [15][16], and such methods could be employed here. However the implementation of these methods is non-trivial and believed to be a research issue on its own right from a pure computational geometry point of view, which is beyond the scope of this paper.

## 6 GENERATING GLOBAL ACCESSIBILITY CONES USING A FIXED-RESOLUTION PARTITIONING OF UNIT SPHERE

This section describes an approach for computing global accessibility cones by introducing approximation and eliminating numerical difficulties. This approach is relatively easy to implement.

## 6.1 Overview of Approach

This approach follows the similar “initialize-update” scheme as in the exact adaptive approach. The difference is that, instead of adaptively refining the partitioning resolution on the unit sphere, a fixed resolution is adopted in the following two steps:

- 1) *Initialize*: In this step we partition the boundary of the unit sphere into spherical triangles with a pre-defined resolution, and initialize the *AccessStatus* matrix. Section 6.2 describes this step in detail.
- 2) *Update*: In this step we update the accessibility information in the *AccessStatus* matrix. Section 6.3 describes an algorithm UPDATE for this step. We calculate inaccessibility region  $I$  by performing inaccessibility test for each pair of non-convex-hull facets that belong to the same concave regions and face each other (property 5 and 6). For each inaccessibility region  $I$ , we perform occupancy test and update the accessibility information for all spherical triangles that lie inside or intersect  $I$ .

This fixed-resolution approach provides conservative approximation. Although for some directions, it may report an accessible facet as inaccessible, but it never reports an inaccessible facet as accessible. That is, it does error only on the safe side. It ensures that accessibility-based decomposition guided by the information from the result of this approach, does not result in any components that still contain inaccessible facets due to the imprecise input.

## 6.2 Initializing Accessibility Matrix

Algorithm INITIALIZE takes the set of non-convex-hull facets  $F_n$  of the object as an argument. The boundary of the unit sphere is partitioned into spherical triangles with a certain pre-defined resolution. The initial accessibility information is stored in matrix *AccessStatus* and will be updated later by algorithm UPDATE.

**Algorithm** INITIALIZE ( $F_n$ )

- 1) Partition the boundary of the unit sphere into spherical triangles.
- 2) Create an  $M \times N$  matrix *AccessStatus*, where  $N$  is the number of the facets in  $F_n$ ,  $M$  is the number of spherical triangles in  $S$ . Each entry of the matrix will indicate the accessibility status of a facet  $f$  from a set of directions defined by a spherical triangle  $s$ . Let  $H(f)$  be the hemisphere with the direction normal of facet  $f$  as its pole. Initially, for a facet  $f$ , if a spherical triangle  $s$  is completely contained by hemisphere  $H(f)$ , the corresponding entry in the matrix is set to TRUE; otherwise, the corresponding entry is set to FALSE.

Note that this initialization is conducted conservatively – an entry associated with a spherical triangle intersecting  $H(f)$  is set to FALSE because facet  $f$  cannot be accessible from the entire spherical triangle.

## 6.3 Updating Initial Accessibility Matrix

To determine the accessibility cone of a non-convex-hull facet  $f$ , it is sufficient to only examine the influence of other non-convex-hull facets facing  $f$  and present in the same concave region. Algorithm UPDATE takes two arguments,  $S$  and  $F_n$ .  $S$  is the set of spherical triangles on the boundary of the unit sphere generated by INITIALIZE.  $F_n$  is the set of non-convex-hull facets on the object.

For each pair of facets  $(f, f')$  in  $F_n$  that face each other and are present in the same concave region, UPDATE calls Algorithm INACCESSIBLE to find the inaccessibility region of  $f$  on the spherical map due to  $f'$ . Figure 3 schematically shows an inaccessibility region overlaid on the set of spherical triangles. Then, it performs occupancy tests to update the matrix *AccessStatus*.

**Algorithm** UPDATE ( $S, F_n$ )

For every pair of facets  $(f, f')$  in  $F_n$  that face each other and are present in the same concave region, do the following:

- 1) *Inaccessibility Test*: Call INACCESSIBLE to calculate the inaccessibility region  $I$  of  $f$  due to the presence of  $f'$ .
- 2) *Occupancy Test*: For every spherical triangle on the unit sphere, OCCUPANCY checks whether it is contained inside the inaccessibility region  $I$ . All the spherical triangles that are contained inside the inaccessibility region  $I$  are marked as inaccessible. To make the algorithm conservative, all spherical triangles that are intersecting the inaccessibility region  $I$  are also marked as inaccessible. The index of facet  $f'$  is added into the entries in the *AccessStatus* matrix corresponding to facet  $f$  and these spherical triangles.

Algorithm UPDATE has the time complexity of  $O(N^2M)$  in the worst case, where  $N$  is the total number of non-convex-hull facets, and  $M$  is the number of spherical triangles. However this is a loose-bound complexity. This algorithm can be made efficient by pruning out unnecessary inaccessibility and occupancy tests. Unnecessary inaccessibility tests have already been pruned out by not doing the inaccessibility tests for facet pairs that are not present in the same concave region and are not facing each other (property 5 and 6).

After an inaccessibility region is calculated, it is sufficient to apply the occupancy tests only for *candidate* spherical triangles. Take the *enclosing spherical rectangle* of the convex inaccessibility region  $I$  and expand it by the resolution of the spherical triangles. This expanded enclosing spherical rectangle is defined as a *candidate patch*. A spherical rectangle can be completely specified by the spherical coordinates of its vertices. The spherical coordinates of a point  $p(\theta, \phi)$  are shown in Figure 4, where  $\theta \in [0, \pi]$  and  $\phi \in [0, 2\pi]$ . All the spherical triangles with vertices inside this candidate patch will be the candidate triangles for occupancy test. The reason for expanding the enclosing spherical rectangle into the candidate patch is to include those spherical triangles that intersect the rectangle but whose vertices do not lie inside the rectangle. Figure 5 shows an enclosing spherical rectangle for an inaccessibility region before expansion. Notice that to calculate  $\theta$ -coordinate extremes, extreme edges have to be found, while the  $\phi$ -coordinate extreme can be calculated by finding the extreme vertices.

A variation of 2-Dimensional *Orthogonal Range Tree* [17] is used to store and query for the vertices of spherical triangles. Let  $Q$  be the set of  $m$  points (vertices of spherical triangles) on the sphere. Range tree  $T$  is a two-level data structure. Figure 6 shows the structure of a range tree.

- The main tree is a balanced binary search tree  $T$  built on the  $\theta$ -coordinate of the points in  $Q$ . The points are stored in the leaves of this main tree.
- For any internal or leaf node  $v$  in  $T$ , the *canonical subset*  $Q(v)$  is stored in a balanced binary search tree  $T_{assoc}(v)$  on the  $\phi$ -coordinate of the points. The node  $v$  stores a pointer to the root of  $T_{assoc}(v)$ , which is called the *associated structure* of  $v$ .

The canonical subset  $Q(v)$  is the subset of points stored in the leaves of the sub-tree rooted at node  $v$ . For instance, the canonical subset of the root of the tree is the whole set  $Q$ . The canonical subset of a leaf is simply the point stored at that leaf. A rectangle query range on  $Q$  asks for the points from  $Q$  lying inside a spherical query rectangle  $[\theta_1 : \theta_2] \times [\phi_1 : \phi_2]$ . A point  $p = (\theta_p, \phi_p)$  lies inside the spherical query rectangle if and only if  $\theta_p \in [\theta_1 : \theta_2]$  and  $\phi_p \in [\phi_1 : \phi_2]$ . Construction time for range tree is  $O(m \log m)$  while the query time is  $O(\log^2 m + k)$ , where  $m$  is the total number of points in  $Q$  and  $k$  is the number of reported points [17]. All the reported points are candidate vertices, which are then tested if they are contained inside the inaccessibility region. If a vertex is contained inside the inaccessibility region, all the spherical triangles sharing that vertex are marked as inaccessible. If not, all the spherical triangles sharing that vertex are tested for intersection with the inaccessibility region.

Algorithm OCCUPANCY takes the set  $S$  of spherical triangles on the boundary of the unit sphere and the inaccessibility region  $I$  as input, and performs occupancy tests in the following steps.

**Algorithm OCCUPANCY** ( $S, I$ )

- 1) Call CALCULATEENCLOSINGRECTANGLE to calculate enclosing spherical rectangle  $R_0$  for the inaccessibility region  $I$ .
- 2) GENERATE the candidate patch  $R$  by expanding the spherical rectangle  $R_0$ .
- 3) Obtain all the spherical vertices  $V_R$ , contained in  $R$  by querying the range tree.
- 4) Call UPDATEACCESSIBILITYMATRIX to update the accessibility matrix *AccessStatus* by testing all the candidate vertices  $V_R$ , and the associated spherical triangles  $T$  for occupancy.

Algorithm CALCULATEENCLOSINGRECTANGLE takes the inaccessibility region  $I$  as input and generates the enclosing spherical rectangle  $R_0$  which is specified as  $[\theta_1 : \theta_2] \times [\phi_1 : \phi_2]$ . This algorithm is straightforward except for three special cases - one of the poles lies inside  $I$ , or arc  $\phi = 0$  intersects  $I$ .

**Algorithm CALCULATEENCLOSINGRECTANGLE** ( $I$ )

- 1) If north pole  $(0, 0)$  is contained in  $I$  (see Figure 7), then  $[\theta_1 : \theta_2] = [0, \theta_{\max}]$  and  $[\phi_1 : \phi_2] = [0, 2\pi]$ .
- 2) Else if south pole  $(\pi, 0)$  is contained in  $I$ , then  $[\theta_1 : \theta_2] = [\theta_{\min}, \pi]$  and  $[\phi_1 : \phi_2] = [0, 2\pi]$ .
- 3) Else if arc  $\phi = 0$  intersects  $I$  (see Figure 8), then, due to discontinuity of  $\phi$  at 0, there will be two spherical rectangles in this case  $[\theta_1 : \theta_2] \times [\phi_1 : \phi_2]$  and  $[\theta_1' : \theta_2'] \times [\phi_1' : \phi_2']$ , where  $[\theta_1 : \theta_2] = [\theta_{\min}, \theta_{\max}]$ ,  $[\phi_1 : \phi_2] = [0, \phi_{\text{lower}}]$  and  $[\theta_1' : \theta_2'] = [\theta_{\min}, \theta_{\max}]$ ,  $[\phi_1' : \phi_2'] = [\phi_{\text{higher}}, 2\pi]$ .
- 4) Else (see Figure 5),  $[\theta_1 : \theta_2] = [\theta_{\min}, \theta_{\max}]$  and  $[\phi_1 : \phi_2] = [\phi_{\min}, \phi_{\max}]$ .

$\theta_{\min}$  and  $\theta_{\max}$  are the  $\theta$ -coordinate extremes of  $I$ , while  $\phi_{\min}$  and  $\phi_{\max}$  are the  $\phi$ -coordinate extremes of  $I$ . Due to discontinuity of  $\phi$  at 0, search will be done in two steps in the above third case.  $\phi_{\text{lower}}$  is the maximum  $\phi$ -coordinate of  $I$  less than  $\pi$ .  $\phi_{\text{higher}}$  is the minimum  $\phi$ -coordinate of  $I$  greater than  $\pi$ . Maximum and minimum  $\phi$  can be computed by finding the extreme vertices of  $I$ . Maximum and minimum  $\theta$  will have to be computed by finding the extreme edges of  $I$ .

Algorithm UPDATEACCESSIBILITYMATRIX takes the set  $S$  of spherical triangles on the boundary of the unit sphere, the inaccessibility region  $I$ , and the set of vertices  $V_R$  as input, and updates the

accessibility matrix.  $V_R$  is the set of vertices reported by the range tree query about the candidate patch  $R$ .

**Algorithm** UPDATEACCESSIBILITYMATRIX ( $S, I, V_R$ )

For every vertex  $v$  in  $V_R$ , do the following:

- 1) Let  $T_v$  be the set of all the spherical triangles sharing  $v$ .
- 2) If  $v$  lies inside  $I$ , then mark all spherical triangles in  $T_v$  as inaccessible and update the corresponding entries in the matrix *AccessStatus*.
- 3) Else, for every spherical triangle  $t$  in  $T_v$ , do the following:
  - If  $t$  intersects  $I$ , then mark  $t$  as inaccessible and update the corresponding entry in the matrix *AccessStatus*.

## 6.4 Implementation and Results

Above-described algorithms have been implemented on Windows XP (Pentium 4 CPU 2.40 GHz and 1 GB RAM) using Visual C++ and ACIS 7. Figure 9, 10, 11 and 12 show four objects that were tested on the system. The number of spherical facets used in all cases is 2700. We created multiple versions of each object by faceting them with different surface tolerance. Surface tolerance is the maximum distance between a facet and the true surface. Table 1 shows the number of facets in each object and the time required to construct the entire *AccessStatus* matrix. This matrix needs to be built only once and can even be built offline and stored with the object model. Once it has been built, it can be used for various application queries. Since these queries are simple mappings and lookups, they are very fast (see Section 7 for details).

Table 1: Implementation results

Part	Number of Facets	Execution Time (seconds)
Object A (Figure 9) 5 Concave Regions	1028	4
	2140	12
	3084	22
	4076	34
Object B (Figure 10) 3 Concave Regions	1016	2
	2028	5
	3120	13
	4004	20
Object C (Figure 11) 1 Concave Region	1024	3
	2008	9
	3050	21
	4020	35
Object D (Figure 12) 1 Concave Region	1064	13
	2090	36
	3084	73
	4052	117

## 6.5 Computing Exact Global Accessibility Cones for a Single Facet

This section a method to compute exact global accessibility cones for a single facet by utilizing the information in the accessibility matrix generated by the approach described in section 6.1. In this approach, a spherical triangle will be marked accessible only if the facet is accessible from

the whole triangle. A spherical triangle will be marked as inaccessible even if it offers partial accessibility. To extract the accessible portions from such spherical triangles, the information stored in the matrix can be used. If a facet is inaccessible from a certain spherical triangle, facet indices of all the blocking facets are also stored in the matrix. These blocking facets can be projected onto the unit sphere to find the exact inaccessibility regions due to these facets. These exact inaccessibility regions can then be subtracted from the spherical triangle to produce the portion of the spherical triangle from which the facet is exactly accessible. The spherical triangles and the inaccessibility regions can also be projected onto a plane using central projection for using the Boolean operations implemented for planar polygons. After performing the Boolean operation, the planar polygon can be converted back to spherical polygon.

## 7 QUERYING ACCESSIBILITY MATRIX

Based on the application of accessibility analysis in the areas of process planning, inspection planning and mold design, following types of querying can be performed on the global accessibility cones represented by the matrix *AccessStatus*.

- 1) *Find the global accessibility cone of a facet.* Given a particular facet on the object, this query finds its facet index and looks up the corresponding column in *AccessStatus*. The rows with TRUE entries corresponding to this column indicate the spherical triangles from which this facet is accessible. The set of all such spherical triangles forms the global accessibility cone of the facet.
- 2) *Find the facets that are accessible from a particular direction.* Given a particular direction as a unit vector, we locate the spherical triangle that contains this direction on the boundary of the unit sphere. The query looks up the row in *AccessStatus* matrix corresponding to the index of the spherical triangle. The columns with TRUE entries corresponding to this row indicate the facets that are accessible from the given direction.
- 3) *Find the facets that are blocking a particular facet.* Given a particular facet on the object, this query looks up the column in *AccessStatus* matrix corresponding to the index of this facet. Each row with the entry of a facet list corresponding to this column gives the indices of the facets that are blocking the given facet in directions represented by a spherical triangle.
- 4) *Compare two or more directions for the amount of accessibility.* For each spherical triangle, number of TRUE entries is added. All the spherical triangles are then ranked based on the sum. Each input direction is mapped to the corresponding spherical triangle and the calculated ranks are returned.

The above queries are simple mappings and lookups. A direction is first mapped to a spherical triangle, then the index of the spherical triangle and the object facet is fetched, and finally, the information stored in the *AccessStatus* matrix at the index is looked up and returned. These queries are very fast to perform and typically take fraction of a second.

## 8 CONCLUSIONS

In this paper we present an algorithm for determining the set of directions from which a triangular facet is inaccessible due to another triangular facet. This algorithm corresponds to the exact mathematical definition of semi-infinite inaccessibility region and is easy to implement. We present two different approaches to compute the global accessibility cone for every facet of an object based on this algorithm. We report detailed experimental results for the approach

described in Section 6 to demonstrate the running performance. The performance is impressive even on a low end PC. Accessibility analysis results for all objects were computed in less than two minutes. The longest time was 117 seconds for an object with 4052 facets with the surface tolerance of 0.025 mm.

Accessibility analysis approach described in this paper presents an improvement over previous approaches in the following aspects:

- A provably sound algorithm has been developed for computing the exact inaccessibility region for a facet due to the presence of another facet.
- Based on the above algorithm, both exact and approximate approaches have been developed to compute the accessibility cones for polyhedral objects. The approximate approach is conservative in nature - it makes errors on the safer side and is guaranteed to correctly identify *all* inaccessible facets on the boundary of the object.

We believe that the algorithms presented in this paper can be used in a wide variety of manufacturing planning applications. Specifically, we are using the method described in this paper in the area of mold design and setup planning for CNC machining.

The accessibility analysis approach described in this paper currently addresses the exact accessibility cones only for polyhedral objects. This approach can be extended by allowing direct projection of curved faces. Further theoretical work will be needed in this area to identify the projection scheme.

**Acknowledgement.** This research has been supported by NSF grant DMI0093142 and ONR grant N000140010416. Opinions expressed in this paper are those of authors and do not necessarily reflect opinion of the sponsors.

## 9 REFERENCES

- [1] Kang, J.K. and Suh, S.H. Machinability and set-up orientation for five-axis numerically controlled machining of free surfaces. *International Journal of Advanced Manufacturing Technology*, 1997, 13: 311-325.
- [2] Suh, S.H. and Kang, J.K. Process planning for multi-axis NC machining of free surfaces. *Internal Journal of Production Research*, 1995, Vol. 33, No. 10, 2723-2738.
- [3] Balasubramaniam, M., Laxmiprasad, P., Sarma, S., and Shaikh, Z. Generating 5-axis NC roughing paths directly from a tessellated representation. *Computer-Aided Design*, 32:261-277, 2000.
- [4] Chen, L.L., Chou, S.Y., and Woo, T. C. Parting directions for mould and die design. *Computer-Aided Design*, 1993, Vol. 25, 763-767.
- [5] S. Dhaliwal, S.K. Gupta, J. Huang, and M. Kumar. A feature based approach to automated design of multi-piece sacrificial molds. *ASME's Journal of Computing and Information Science in Engineering*, 1(3):225--234, 2001.
- [6] Weinstein, M. and Manoochchri, S. Optimum parting line design of molded and cast parts for manufacturability. *Journal of Manufacturing Systems*, 1997, Vol. 16, No. 1, 1-11.
- [7] Spitz, S. N., Spyridi, A. J., and Requicha, A. A. G. Accessibility analysis for planning of dimensional inspection with coordinate measuring machines. *IEEE Transactions on Robotics and Automation* v 15 n 4 1999 IEEE p 714-727 1042-296X.

- [8] Woo, T. C. Visibility maps and spherical algorithms. *Computer-Aided Design*, 1994, Vol. 26, No. 1, 6-1.
- [9] Kim, D.S., Papalambros, P. Y., and Woo, T. C. Tangent, normal, and visibility cones on Bézier surfaces. *Computer Aided Geometric Design*, 1995, Vol. 12, 305-320.
- [10] Elber, G. and Cohen, E. Arbitrarily precise computation of Gauss maps and visibility sets for freeform surfaces. In *Proceedings of 3<sup>th</sup> Symposium on Solid Modeling and Applications*, May 1995, Salt Lake City, Utah.
- [11] Stage, R. and Roberts, C. A framework for representing and computing tool accessibility. *Proceedings of DETC'97*, September 14-17, 1997, Sacramento, California.
- [12] Lenthem, J. G. *Spherical trigonometry, for the use of colleges and schools*, London, Macmillan, 1949.
- [13] Chen, L.-L., and Woo, T. C. Computational geometry on the sphere with application to automated machining. *Journal of Mechanical Design*, 1992, Vol. 114, 288-295.
- [14] Lo, S. H. Delaunay triangulation of non-convex planar domains. *International Journal for Numerical Methods in Engineering*, vol. 28, 2695-2707 (1989).
- [15] Hoffmann, C., Hopcroft, J., and Karasick, M. Towards implementing robust geometric computations. In *4<sup>th</sup> Annual ACM Symposium of Computational Geometry*, pp106-117, 1988.
- [16] Sahni, S. *Data Structures, Algorithms, and Applications in C++*, McGraw-Hill, 1998.
- [17] Berg, M. de, Kreveld, M. van, Overmars, M., and Schwarzkopf, O. *Computational Geometry: Algorithm and Applications*, Springer-Verlag, 2000.

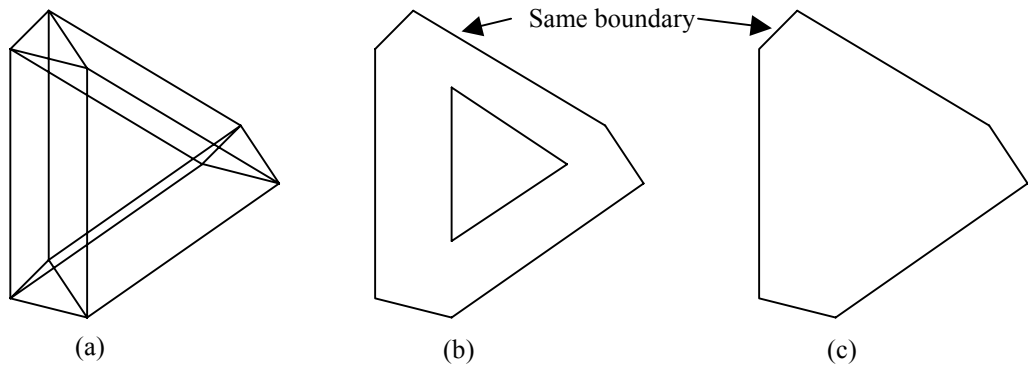


Figure 1: Planar representation of  $I$ ; (a) Projection of facet  $f'$  on unit sphere as it moves along edges of facet  $f$ ; (b) Boundaries of the projection; (c) The convex inaccessibility region

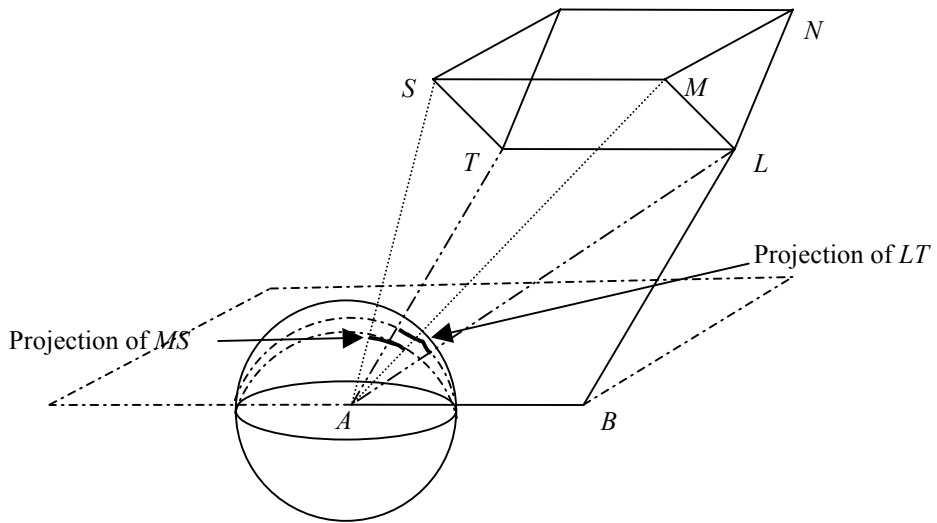


Figure 2: Projection of lines  $MS$  and  $LT$  on sphere

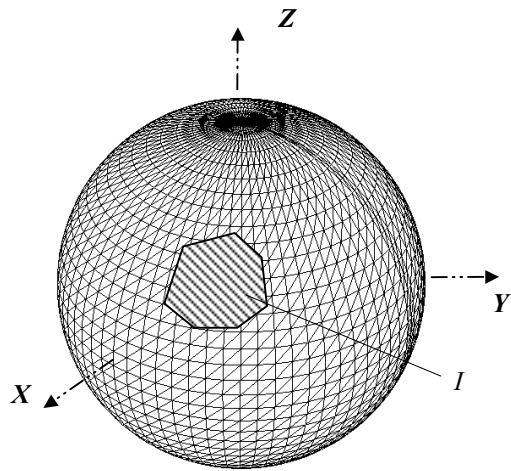


Figure 3: Inaccessibility region overlaid on set of spherical triangles

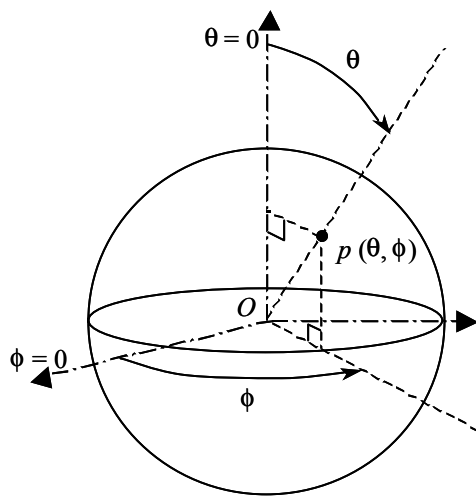


Figure 4: Spherical coordinates of a point  $p(\theta, \phi)$

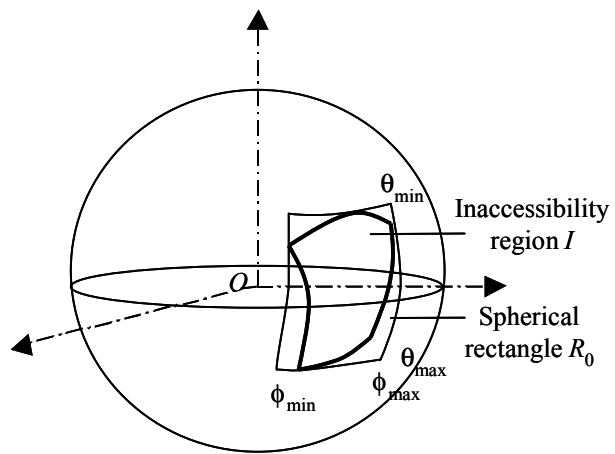


Figure 5: Inaccessible region  $I$  and enclosing spherical rectangle  $R_0$

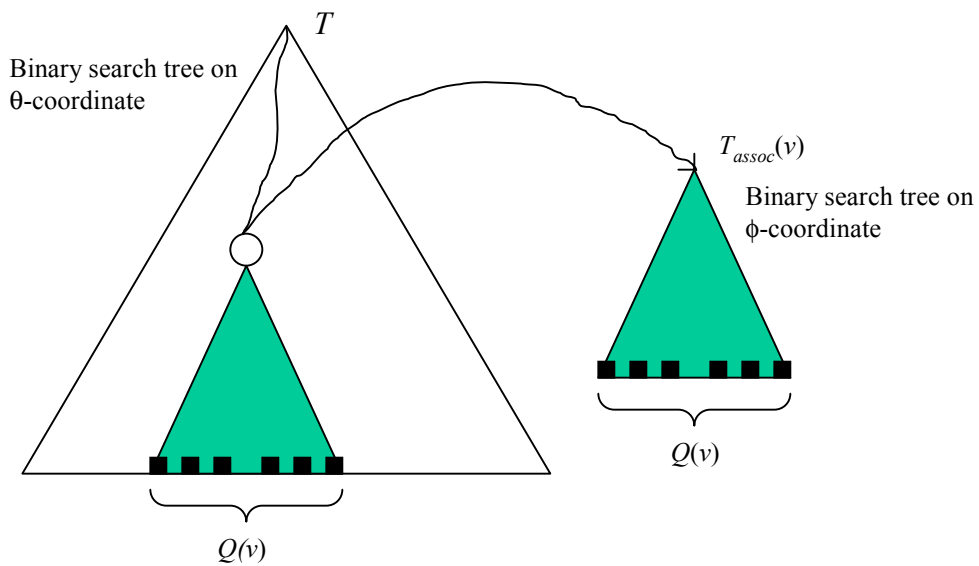


Figure 6: 2-Dimensional Orthogonal Range Tree

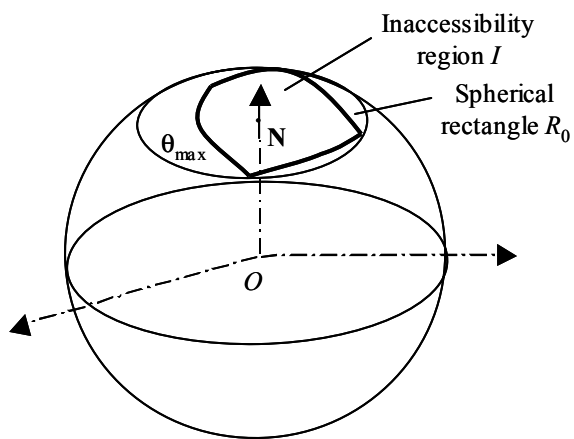


Figure 7: Enclosing spherical rectangle  $R_0$  when inaccessibility region contains the North pole

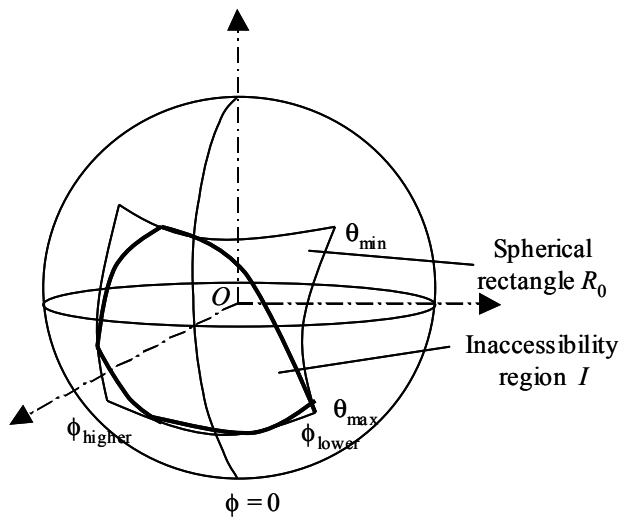


Figure 8: Enclosing spherical rectangle  $R_0$  when arc  $\phi = 0$  intersects inaccessibility region

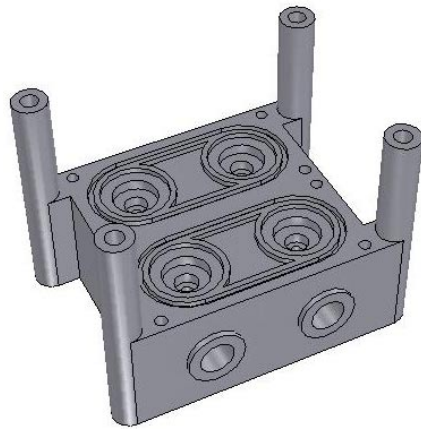


Figure 9: Object *A*

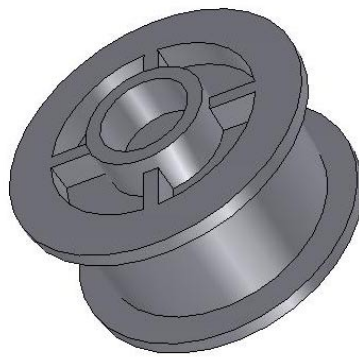


Figure 10: Object *B*

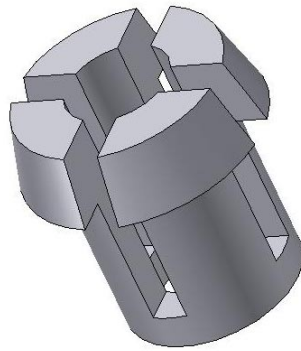


Figure 11: Object *C*

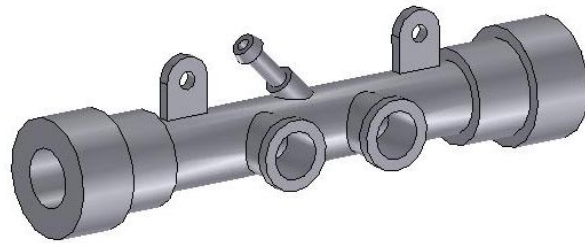


Figure 12: Object *D*