COMPUTATIONAL GEOMETRY ON THE SPHERE WITH APPLICATION TO AUTOMATED MACHINING

Lin-Lin Chen
Tony C. Woo

Department of Industrial and Operations Engineering
The University of Michigan
Ann Arbor, MI 48109-2117

Technical Report 89-30

August 1989

COMPUTATIONAL GEOMETRY ON THE SPHERE WITH APPLICATION TO AUTOMATED MACHINING

Lin-Lin Chen
Tony C. Woo

Department of Industrial and Operations Engineering
The University of Michigan
Ann Arbor, Michigan 48105-2117

August 1989

Abstract

Based on observations made on the geometry of the cutting tools and the degrees of freedom in 3-, 4-, 5-axis numerical control machines, a new class of geometric algorithms is induced – on the unit Sphere. Spherical algorithms are useful for determining the type of tool, its path, workpiece fixturing, and the type of machine.

Basic to these algorithms are four that are presented here: detection of convexity on the sphere, computation for spherical convex hull, determination of the spherical convexity of a union and the intersection of hemispheres. These four algorithms are related by duality and the sharing of partial results.

1. Introduction

The term "computational geometry" has two interpretations. In his address to the Royal Society, Robin Forrest [4] used the term in the context of computations on *analytically* complex surfaces for geometric design. As *continuity* is important to the design of complex surfaces, tangents, normals, and their (dot and cross) products are basic to the computation. The other interpretation stems from an almost orthogonal field in geometry – one that is *combinatorially* complex. Shamos [16], in his celebrated dissertation, gave a procedural treatise of classical geometric constructs, such as the convex hull. Notably, in the *discrete* domain of vertices, edges, and polygons, he injected the rigor of algorithm analysis.

This paper is an attempt to unify these two fields of research by addressing the algorithmic issues of analytically complex geometric entities. Its roots are humble – numerical control (NC) machining and workpiece setup. But one of the offsprings is rather pleasant – geometric algorithms on the unit sphere, or spherical algorithms, the basics of which are presented here.

The applications for which the algorithms are developed involves k-axis NC machines, where k = 3, 4, or 5. Given an object represented by N free-form surfaces, NC applications seek the number of setups (i.e. orientations of the workpieces) and for each setup the tool path. While, it is customary to compute cutter orientations from surface normals, there are two hidden implications. First, the type of cutter is already fixed. If the orientation of the cutter is set to the normal (with the possible difference in the sign of the vectors), the most "efficient" cutter is the flat-end mill† . Second, the type of NC machine is also pre-determined. For most free-form surfaces, the wide range of their surface normals can be accommodated only by a 5-axis NC machine. Now, it is not unreasonable to ask the question: "Can a 3-axis (or a 4-axis) NC machine with a different cutter accomplish the same task?" To

† If the cutter orientation is orthogonal to the normal, a side mill is implied.

be sure, a 5-axis NC machine is more flexible, but is also much more expensive (hence less available) than a 3- or 4-axis machine. Furthermore, the tool path generated for a 3-axis machine is valid for a 4- or 5-axis machine but not vice versa. These observations lead to the following problem definition.

Problem MS: Machine Selection. Given a workpiece consisting of N free-form surfaces, determine the minimum number of axes required of the NC machine and the type of cutter.

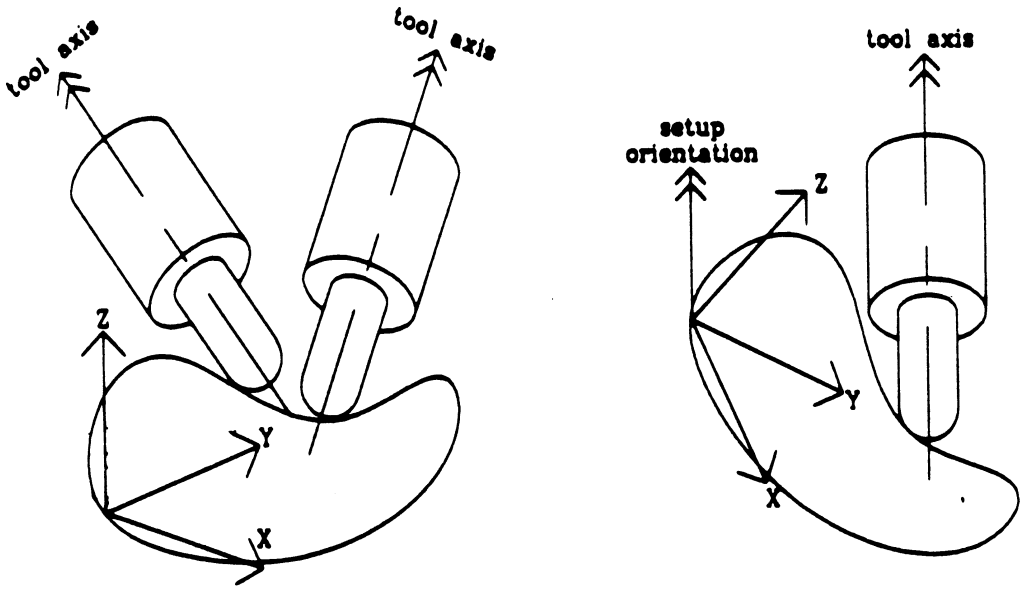
Problem MS as formulated is not without a caveat. It is intuitive that more setups would be required of a 3-axis machine than that of a 4- or 5-axis machine as illustrated in Figure 1.1. Since mounting and dismounting a workpiece is not considered to be productive, it is therefore useful to able to cut as many surfaces as possible in any single setup. This induce a companion problem definition:

Problem WS: Workpiece Setup. Given a workpiece consisting of N free-form surfaces, determine an setup orientation such that the maximum number of surfaces are accessible.

<Insert Figure 1.1>

One of the useful notions from computational geometry is visibility. Two points are visible to each other if the line segment joining them has no intersection [17]. For each point on the surface, it is possible to compute visible rays (to points at infinity). The intersection of the rays for all points on the surface forms a *visibility cone* for the surface. Thus, if the workpiece is oriented such that the axis of the cutting tool lies in the same direction as one of the rays in the visibility cone, the corresponding surfaces are accessible for machining.

Now, consider the geometry of a cutting tool as related to the geometry of a surface. The directly available information from a point on a surface is the



(a) 5-axis machining

(b) 3-axis machining

Figure 1.1 Machine Selection and Workpiece Setup.

outward-pointing normal. Indeed, for a surface, there exists a *Gaussian Map* which is the intersection of the surface normals with the unit sphere [7]. Without loss of generality, assume that there are three kinds of cutting tools for a milling machine: flat-end, semi-ball-end, and ball-end. With respect to a normal (i.e. a point in the Gaussian Map or GMap), the three kinds of cutting tools can deviate by an angle $v = 0$, $0 < v < \pi/2$, and $v = \pi/2$, respectively. This is illustrated in Figure 1.2. Thus, the visibility of a normal is enhanced by an angle of $0 \leq v \leq \pi/2$, depending on the type of cutting tool used. In the subsequent discussion, the ball-end mill, for which $v = \pi/2$, is chosen.

<Insert Figure 1.2>

The visibility map, or VMap, for a given surface can now be constructed. A VMap is an image on the unit sphere whereby every point in the VMap deviates from a corresponding point in a GMap by $v \leq \pi/2$. The construction of a VMap from a GMap is shown in Figure 1.3. For each point p_i in the GMap, there corresponds a hemisphere c_i (because $v = \pi/2$). If the GMap is a single point (e.g. the north pole), the VMap is a hemisphere (e.g. the Northern Hemisphere). If the GMap consists of two distinct points, the VMap is the intersection of the two corresponding hemispheres. While a GMap is not necessarily (spherically) convex, a VMap is guaranteed to be convex, as it is formed by the intersection of hemispheres that are convex. (More rigorous definitions of spherical convexity are given in Section 2.)

<Insert Figure 1.3>

Problem WS can now be addressed. A collection of N surfaces is represented by N VMaps on the unit sphere, some of which may overlap. If two VMaps intersect, there exists a common direction along which both surfaces can be machined in a single setup. The degrees of freedom of the 3-, 4-, 5-axis milling machines (Problem MS) are next related to the VMaps.

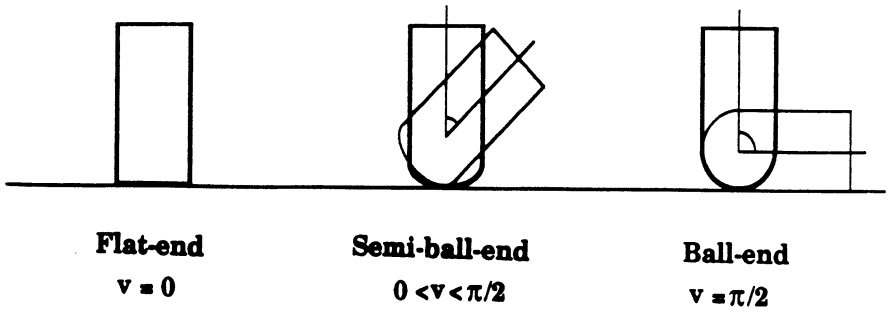


Figure 1.2 The v Angle of a Cutting Tool.

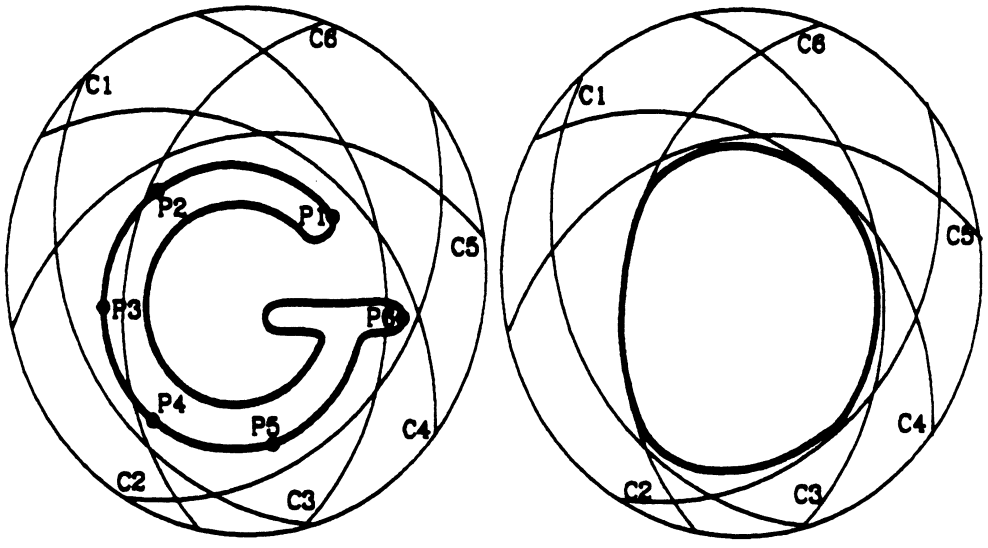


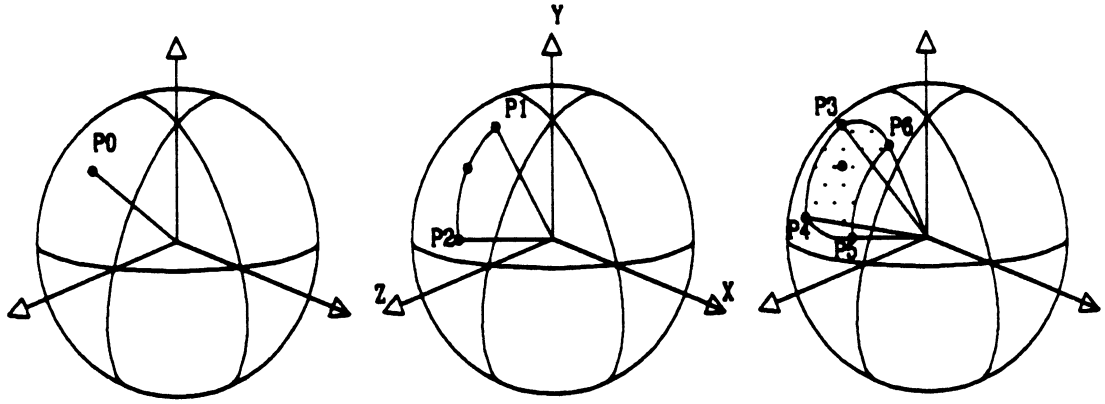
Figure 1.3 Construction of a Visibility Map.

A 4-axis machine is conceptually a 3-axis machine with a rotating table. This extra degree of freedom in rotation can be represented by a great circle, if the rotation covers 2π; otherwise, it is an arc. This situation is illustrated in Figure 1.4(b). For a single setup, if the given arc (great circle) passes through a VMap, the corresponding surface is machinable. The accessibility of surfaces for a 4-axis machine can therefore be formulated as follows: Given N VMaps on the unit sphere, determine an orientation such that a given arc (or a great circle) passes through them (maximally). The utility of a 5-axis machine is now easily representable in spherical geometry. The two additional degrees of freedom in orthogonal rotation correspond to a spherical rectangle as illustrated by Figure 1.4(c). Thus, given a 5-axis machine, the setup of the workpiece can be formulated as a covering problem: Given N VMaps, determine an orientation such that a given spherical rectangle covers them (maximally).

<Insert Figure 1.4>

By invoking spherical geometry, the preceding description of NC machining and setup induces a structure as shown in Figure 1.5. In this paper, four basic spherical algorithms are presented. They are (1) the detection of convexity (via hemisphericity), (2) the computation for a (spherical) convex hull, (3) the determination of the convexity of a union, and (4) the determination of the intersection of hemispheres. While these four problems may seem disparate, they are shown to be tightly woven via duality and the sharing of partial results. In this order, the algorithms are given, after a brief introduction of terms in spherical geometry and central projection.

<Insert Figure 1.5>



(a) 3-axis (b) 4-axis (c) 5-axis

Figure 1.4 Spherical representation of NC Machines.

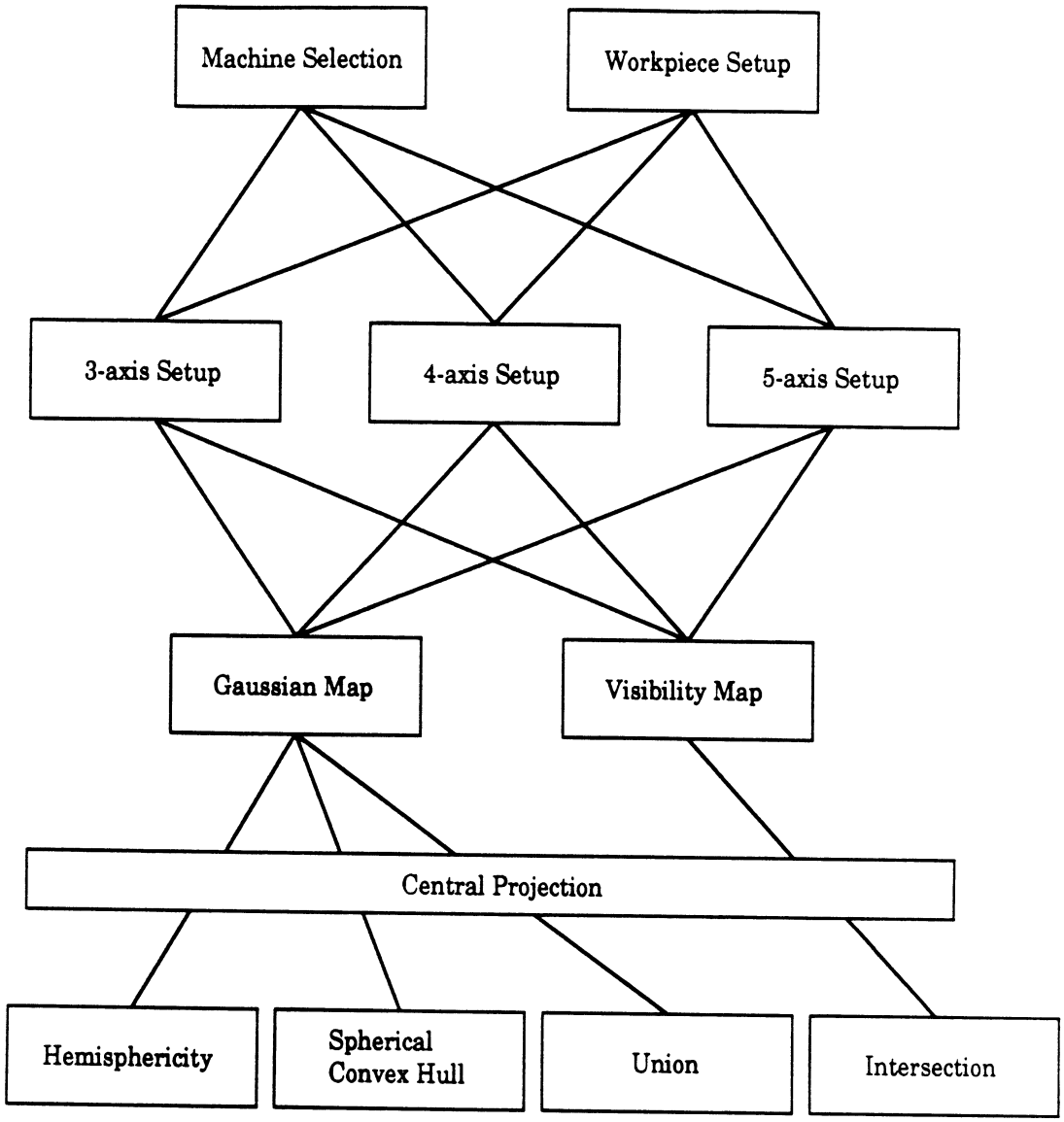


Figure 1.5 Problem Structure

2. Definitions and Representations

In spherical geometry, the objects considered are sets of points on the d dimensional unit sphere S^d. Let E^{d+1} denote the d+1 dimensional Euclidean space and $p = (x_1, x_2, ..., x_{d+1})$ denote a point in E^{d+1}, then

$$S^d = \{ p : |p| = 1 \}$$

In this paper, the discussion will be on S^2, the sphere with radius r = 1 in E^3. The definitions of the principal objects and important concepts are now briefly reviewed [15]:

Point. A point p in S^2 is a unit vector in E^3, which is denoted by a 3-tuple (x_1, x_2, x_3) in this paper. This representation is chosen (over spherical coordinates) such that the correspondence between S^2 and E^2 through projection can be easily maintained, as will be further explained in Section 3.

Antipodal points. Two points p and q are called antipodal if p = -q, where p is called the *antipode* of q and vice versa. A pair of antipodal points divides the infinite number of great circles containing them into semicircles.

Circle and Hemisphere. A circle in S^2 is determined by the intersection of the unit sphere with a plane which meets the sphere at more than one point. If the plane contains the origin, the intersection is a *great circle*; otherwise, it is a *small circle*. (In this paper, the term *circle* is used interchangeably with small circle.) The surface of the sphere is partitioned into two *hemispheres* by a great circle. A *closed hemisphere* includes the bounding great circle, while an *open hemisphere* excludes it. The surface of the sphere is partitioned into two portions by a small circle; the smaller of the two is the *interior* of a small circle. The *center* of a circle is a point in the interior and equidistant from all the points on the circle. A circle is represented by its center and radius in this paper, as its counterpart in the Euclidean plane.

Line. A *line* L in S^2 is a *great circle,* the intersection of the unit sphere with a plane containing the origin. (The term *line* and the term *great circle* are used interchangeably.) Since the radius of a great circle is always equal to $\pi/2$ (as explained under *Distance*), it can be fully represented by the unit normal of the intersection plane.

Line segment. The line segment joining two distinct points p and q in S^2, denoted by pq, is the shorter of the two arcs in the great circle containing p and q. This segment is unique, except for the case when p and q are antipodal points where any great semicircle joining p and q is a segment.

Distance. The distance between two distinct points p and q in S^2 is the length of the line segment (the geodesic) joining p and q. It is defined by the metric:

$$d(p,q) = \cos^{-1}(p \cdot q)$$

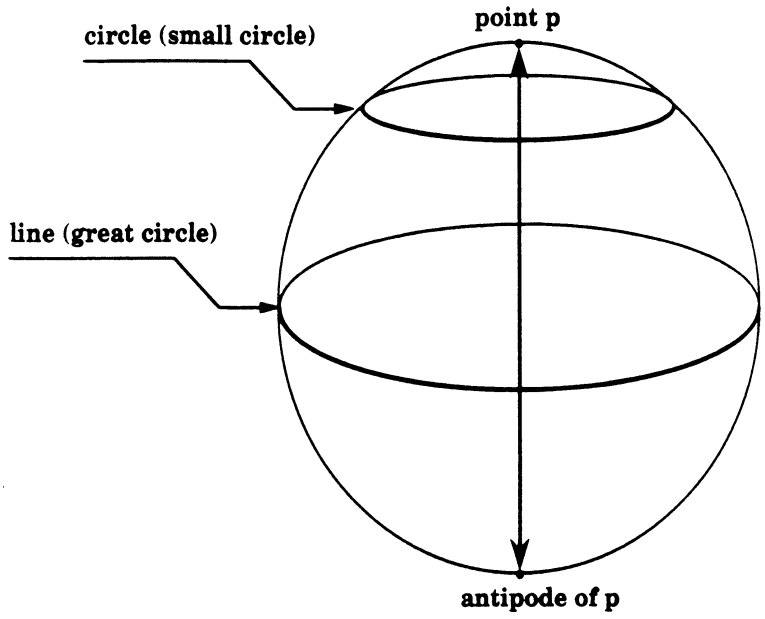
<Insert Figure 2.1>

Polygon. A (spherical) polygon P is a closed path on the sphere of connected ordered line segments ab, bc,..., pq, qa which do not cut across themselves. The points a, b, c, ..., p, q are called the *vertices* of P. The line segments ab, bc, ..., pq, qa are called the *edges* of P.

Hemisphericity. A set of points in S^2 is said to be hemispherical if it is contained in some open hemisphere [1].

Convexity. A set of points P in S^2 is said to be (spherically) *convex*, if for any two points p and q in P, the segment joining p and q lies entirely in P. Notice that, a closed hemisphere contains antipodal points but it does not contain all the semicircles joining them. Therefore, it is not convex.

Quasi-convexity. A set of points P in S^2 is said to be *quasi-convex,* if, for any pair of non-antipodal points p and q in P, the segment pq is also in P. A closed hemisphere is quasi-convex [18].



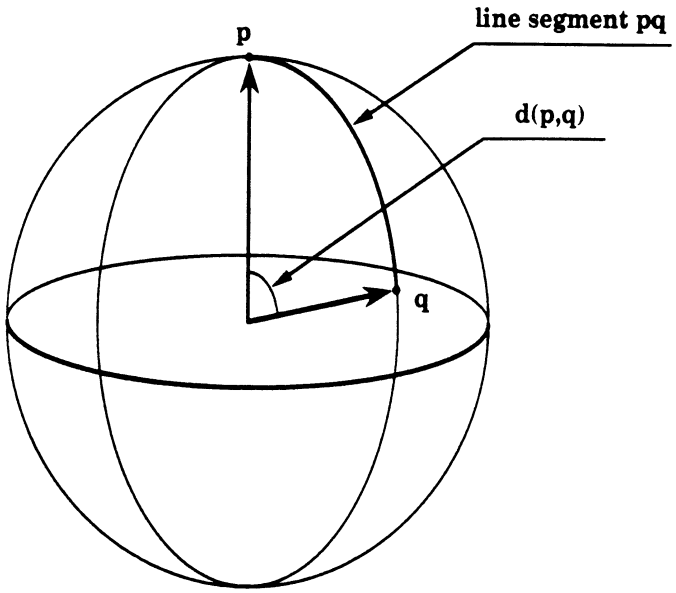


Figure 2.1 Entities on the Sphere

Strict convexity. A set of points P in S^2 is said to be *strictly convex*, if it is nonempty, includes no antipodal pair, and for any two points p and q in P the segment pq is also in P [18].

Convex Hull. A (spherical) convex hull of a set of points P in S^2, denoted by SCH(P), is the boundary of the smallest convex set in S^2 containing P [14].

<Insert Figure 2.2>

3. Projecting onto the Euclidean Plane

Since extensive study in Computational Geometry has been done in 2- and 3-dimensional Euclidean space, it is natural to convert a spherical problem to one in the plane(if possible), apply some planar algorithms, and then convert the planar solution back for the spherical one. One of the methods that converts a spherical problem to a planar one is the *central projection* (gnomonic projection). However, it does not apply in certain cases.

3.1 Central Projection

Central projection maps a pair of antipodal points in S^2 to a point in the two dimensional projective plane P^2, which consists of E^2 augmented by a line at infinity. By emitting a ray from the origin, a point (x_1,x_2,x_3) in S^2 with $x_3 > 0$ is mapped to $(x_1/x_3,x_2/x_3,1)$ in the plane $x_3 = 1$, corresponding to the point $(x_1/x_3,x_2/x_3)$ in P^2. By extending the ray in the opposite direction, $(-x_1,-x_2,-x_3)$, the antipode of the point (x_1,x_2,x_3) is also mapped to $(x_1/x_3,x_2/x_3,1)$ in $x_3 = 1$ (and thus $(x_1/x_3,x_2/x_3)$ in P^2). A point (x_1,x_2,x_3) with $x_3 = 0$ is mapped onto the line at infinity in P^2. Central projection therefore is a two-to-one mapping of points in S^2 (except for points on the equator) to points in E^2. As illustrated in Figure 3.1, a pair of antipodal points, p and

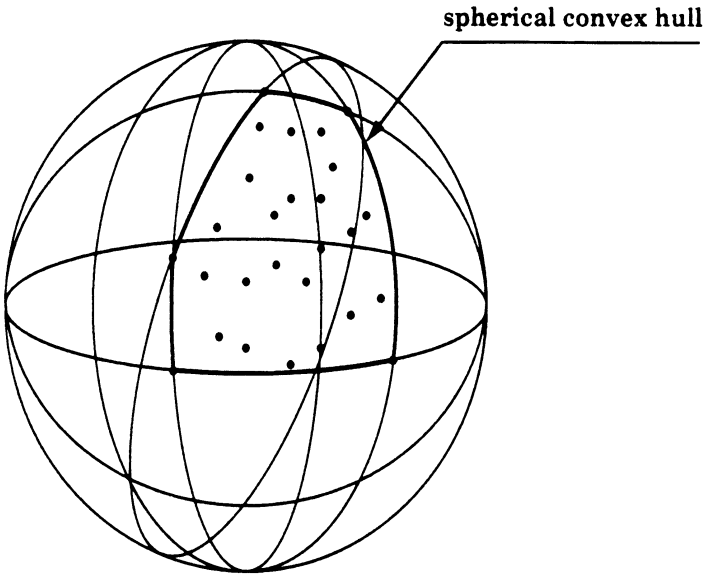


Figure 2.2 Spherical Convexity

-p, in S^2 are mapped to a point p' in E^2, while a great circle is mapped to a line through central projection.

<Insert Figure 3.1>

Since central projection is a two-to-one mapping, by keeping track of the sign of x_3 for a point, it provides an easy way of going back and forth between a geometrical problem in S^2 and a corresponding one in E^2. Nevertheless, central projection is not a panacea: Not all spherical problems can be solved by applying planar algorithms to the projected counterparts. An example is the problem of finding the smallest enclosing circle for a set of points in S^2 because a circle in E^2 may not be mapped to a circle in S^2. It is necessary, then, to review the properties of central projection that allow a spherical problem to be solved in E^2 and the limitations that induce new spherical algorithms.

3.2 Properties of Central Projection

The merit of central projection is the one-to-one and onto mapping between a line in E^2 and a great circle in S^2.

Lemma 3.2.1 *Central projection establishes an one-to-one and onto mapping between great circles in S^2 and lines in E^2, except for the equator, the great circle that lies on the plane $x_3 = 0$.*

Proof. Except the equator, any great circle in S^2 determines a plane through the origin in E^3, which, when intersecting with the plane $x_3 = 1$, determines a line in the plane. Conversely, a line in the plane $x_3 = 1$ together with the origin determine a plane in E^3, which, when intersecting with the unit sphere, determines a great circle in S^2. Notice that the plane $x_3 = 1$ is itself a space E^2. ❑

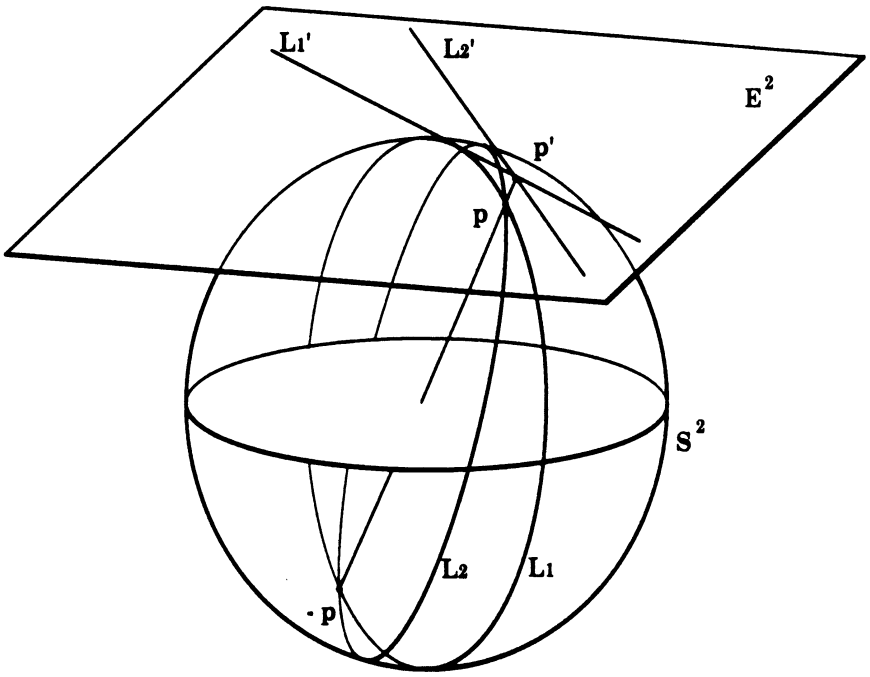


Figure 3.1 **Central Projection**

By Lemma 3.2.1, central projection preserves incidence since a point $p = (x_1,x_2,x_3)$ on a great circle L in S^2 is mapped through central projection to a point $p' = (x_1/x_3,x_2/x_3)$ on L', the unique image of L in E^2. Obviously, central projection also preserves collinearity. Because of the above properties, the intersection of spherical entities (lines, line segments, polygons, etc.) can be obtained by applying central projection and planar algorithms.

Lemma 3.2.2 *Central projection maps a (spherically) convex polygon on S^{2+}, the hemisphere of S^2 corresponding to $x_3 > 0$, to a convex polygon in E^2. Conversely, the inverse of central projection maps a convex polygon in E^2 to a (spherically) convex polygon on S^{2+}.*

Proof. Let P be a convex polygon on S^{2+} and P' be the image of P through central projection. A convex cone in E^3 with the apex at the origin can be constructed by intersecting all the halfspaces each of which is defined by a plane through the origin and an edge of P and contains P. The intersection of this convex cone with the plane $x_3 = 1$ is P', which is convex since it is the intersection of two convex sets in E^3. The converse can be shown similarly by constructing a convex cone in E^3 using the edges of P' and the origin. ❑

As illustrated in Figure 3.2, a spherically convex polygon not crossing the equator is mapped to a convex polygons in E^2 through central projection. On the other hand, a spherical convex polygon crossing the equator is mapped to two open polygons. By Lemma 3.2.2, if a proper rotation is known for a set of points P on S^2 such that, after the rotation, the set will be on S^{2+}, then the convex hull for the set can be constructed by applying planar convex hull algorithm to P', the image of P in E^2 through central projection.

<Insert Figure 3.2>

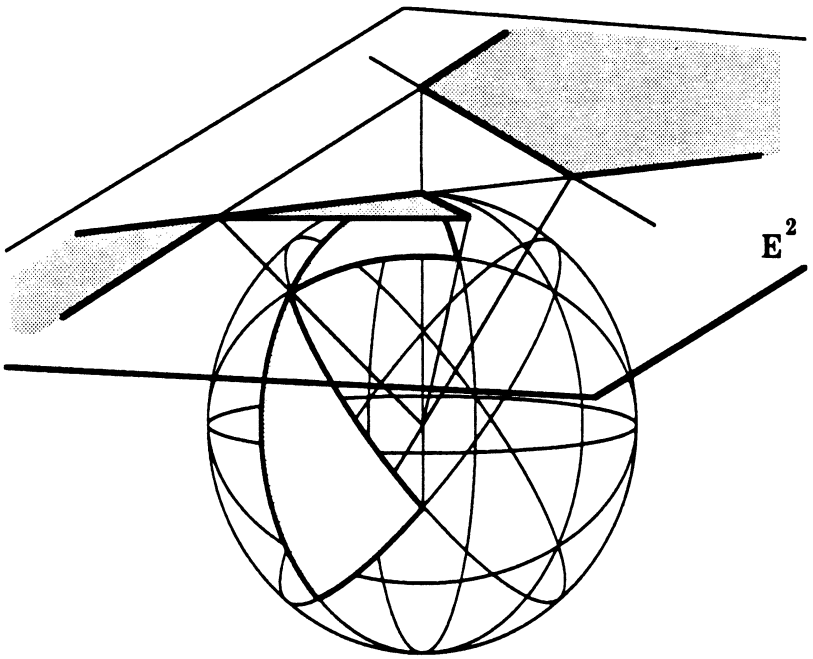


Figure 3.2 **Central Projection of a Polygon**

3.3 Limitations of Central Projection

For certain problems such as finding the smallest enclosing circle for a set of points in S^2, central projection is not applicable because of the intrinsic nature of these problems. In order to apply central projection (or any other projective mapping), some knowledge of the normal of a good projection plane, e.g. (0,0,1) for $x_3 = 1$, is required. Since a circle in E^2 is mapped to a circle in S^2 if and only if the normal of the projection plane is the normal of the plane contains the circle in S^2, finding such a projection and finding the smallest enclosing circle are essentially the same problem that must be solved on the sphere.

Central projection suffers from two limitations. First, central projection does not preserve angle, distance, or area. Therefore, in general, proximity problems, such as the construction of the Voronoi diagram or minimum spanning tree, cannot be solved by applying central projection and planar algorithms.

Second, central projection produces largely distorted images for points close to the equator in S^2. (This is a problem that needs to be taken care of, because planar algorithms usually do not handle points at infinity.) Hence, it is important to find some initial rotations such that, after applying the rotations to the set of points in S^2, no point lies on the equator and therefore none is projected to infinity. It is also essential to find a projective plane such that all projected points are as far away from infinity as possible, so that large distortions and round-off errors can be avoided.

Problem P (Projective)**: Finding Initial Rotations.** Given a set of N points P in S^2, find some rotations such that, after P is rotated, none of the points in P lies on the equator[††].

[††] It is possible to formulate a problem of finding the optimal initial rotation: Given a set of N points P in S^2, find an optimal rotation such that, after P is rotated, all of the points in P are the farthest away from the equator. This optimal initial rotation can be determined, once the largest empty double wedge on the sphere is found. However,it is shown in [3] that

One such algorithm (a slightly modified version due to different dimensionalities) is by Preparata and Muller [13] which finds an initial rotation R_0 such that, after applying R_0, no point in P is projected to infinity. This algorithm, however, does not try to move the points as far away from the equator as possible.

The algorithm proposed here finds two rotations, R_1 and R_2 that move the points in P as far away from the equator as possible. Let $p_i = (x_{i1}, x_{i2}, x_{i3})$, $i = 1,...,N$, be points in P. The points $p_i = (x_{i1}, x_{i2}, x_{i3})$ are mapped to $p_i' = (0, x_{i2}, x_{i3})$ when parallelly projected onto the plane $x_1 = 0$, itself a space E^2. The unit circle S^1, the image of the unit sphere in the plane $x_1 = 0$, is partitioned into (circular) segments by the unitized p_i' and their antipodes in S^1. Since the partition is symmetrical with respect to (0,0), only the partition on a semicircle needs to be considered. Arbitrarily, let the semicircle be one of the two determined by the unitized p_1' and its antipode. The problem is now transformed to finding the maximum (circular) gap between the points on the semicircle, for which Gonzalez' O(N) algorithm [5, 16] can be used. Once the maximum gap is found, R_1 is a rotation that rotates the circular midpoint of the maximum gap to be on the line $x_3 = 0$ (in the plane $x_1 = 0$). After R_1 is applied to the unit sphere, the points in P are moved away from the equator, except for the fixed points (1,0,0) and (-1,0,0). Let P' be the result of applying R_1 to P. After points in P' are parallelly projected onto the plane $x_2' = 0$, the same procedure is performed again to obtain R_2.

An illustration of the algorithm is given in Figure 3.3. Given eight points (unit vectors) on the sphere, where point 3, 4, 5, and 6 are on the equator, find a rotation R_1 to move points 5 and 6 away from the equator. After the points are parallelly projected to the $x_1 = 0$ plane, the largest circular gap is found as shown in Figure 3.3(b). R_1 is the transformation that rotates the

---

there are $O(N^2)$ empty double wedges. To find the largest empty double wedge, it requires an enumeration, which takes $O(N^2)$ time.

bisector of the largest gap to the equator. Figure 3.3(c) shows that points 5 and 6 are moved away from the equator after the application of R_1.

<Insert Figure 3.3>

It is noted that, after applying R_1 to the unit sphere, the (1,0,0) and (-1,0,0) are the only points that can possibly lie on the plane $x_3 = 0$. The application of R_2 to the unit sphere serves to move these two points away from the equator. In addition, each rotation locally moves the points as far away from the equator as possible by moving the circular midpoint of the maximum gap to the equator. This algorithm runs in O(N) time, where N is the number of points in P. Since the procedure for finding R_2 is the same as that for R_1, only the latter is described here:

Algorithm P: Initial Rotations

Step 1. Parallelly project the points in P onto the plane $x_1 = 0$. (This can be done in O(N) time.)

Step 2. Find the maximum circular gap.
Let $p_i = (x_{i1},x_{i2},x_{i3})$, i = 1,...,N, be the unit vectors in P and $p_{i'} = (x_{i2},x_{i3})$ be the image of p_i in the plane $x_1 = 0$. Let $p_1' = (x_{12},x_{13})$ be the base vector and the semicircle considered be consisted of all the vectors having a positive cross product with p_1'. Let q_i, i = 1,...,N, represents the degree of the angle between the two vectors p_1' and p_i' (or the its antipode $(-x_{i2},-x_{i3})$ if it has a negative cross product with p_1'). Then $q_1 = 0$. Let q_{N+1} be equal to π. Now Gonzalez's algorithm can be applied to q_i, i = 1,...,N+1, to find the largest gap and its two end points, q_j and q_k. (This can be done in O(N) time.)

Step 3. Compute the locally optimal rotation R^1.
Let q_0 be the angle between p_1' and (1,0), a unit vector on $x_3 = 0$. If (1,0) has a positive cross product with p_1', R_1 consists of a rotation of $-q_0 + (q_j+q_k)/2$; otherwise, R_1 consists of a rotation of $q_0 + (q_j+q_k)/2$. (This can be done in O(1) time.)

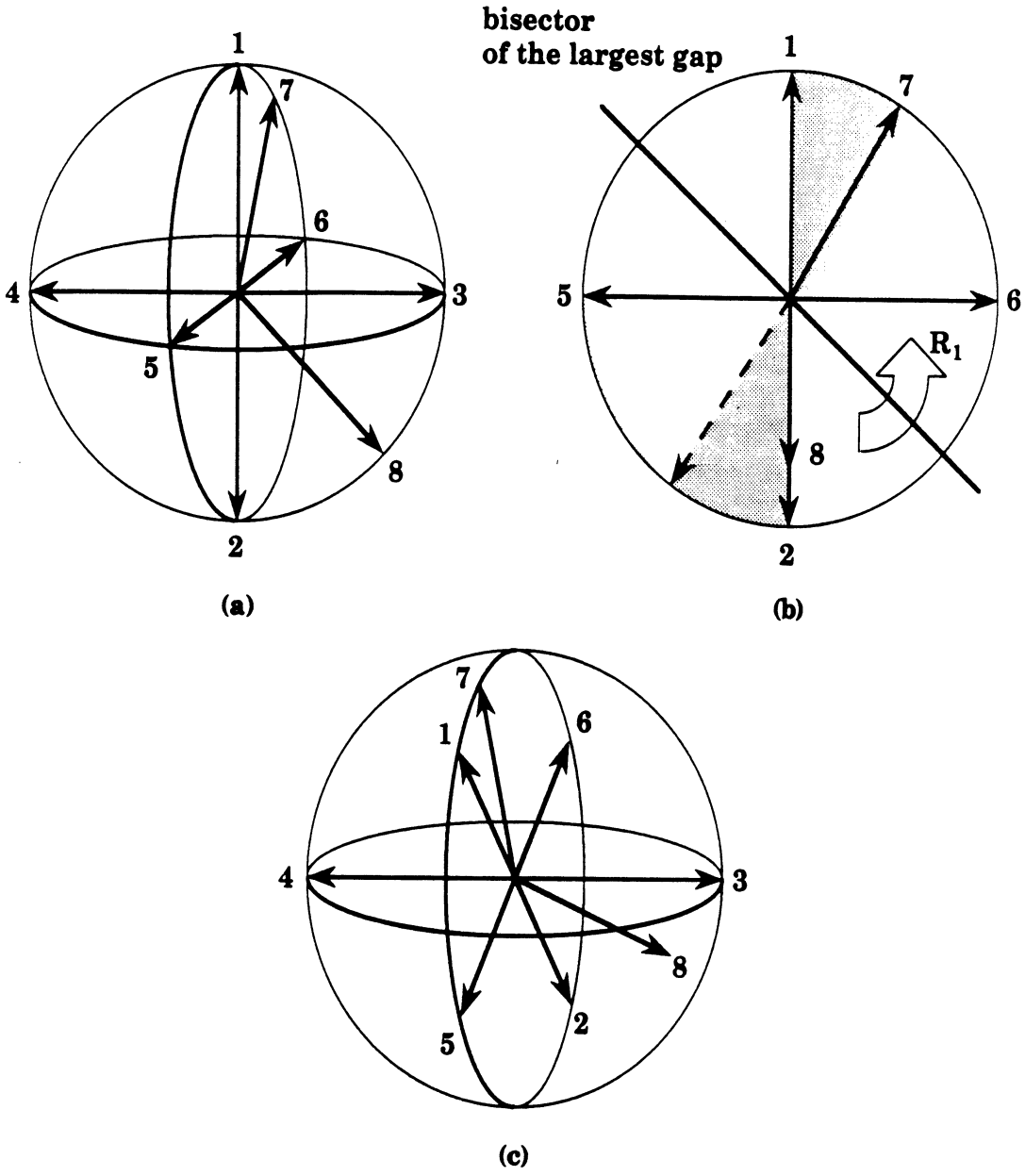


Figure 3.3 Rotating Points Away From the Equator

Step 4. Apply R_1 to P. (This can be done in O(N) time.)

4. Algorithms

Several basic algorithms for solving geometrical problems on the unit sphere are given. They are useful when applied to determining the orientations of the workpiece in NC machining.

Problem S1 (Spherical)**: Detecting Hemisphericity**. Given a set P of N points in S^2, determine whether P is hemispherical (contained in an open hemisphere.) If P is hemispherical, find a great circle which determines a hemisphere that completely contains P.

The detection of hemisphericity for a set P of N points in S^2 is basic to spherical algorithms. Since central projection establishes a one-to-one relations between points in S^2 and points in E^2, if P is hemispherical, it is relatively straight forward to convert a spherical problem for P to a planar counterpart. On the other hand, if P is not hemispherical (or, more precisely, if P contains antipodal points), the convex hull for P is the entire sphere by the definition of convexity.

Problem S1 can be solved in E^2 through proper transformations. For simplicity, assume that none of the points in P lies on the plane $x_3 = 0$. (If not, Algorithm P described in Section 3.3 can be applied to ensure this.) The points in P can be partitioned into two possibly empty sets P^+, those points with $x_3 > 0$, and P^-, those with $x_3 < 0$. If either of the two sets is empty, Problem S1 is trivial; P is hemispherical and is contained in a hemisphere determined by the great circle on $x_3 = 0$. If both sets are not empty, P^+ and P^- are mapped to P'^+ and P'^- in E^2, respectively, through central projection. Planar algorithms can now be applied to find the convex hulls of the two planar sets, $CH(P'^+)$ and $CH(P'^-)$. Given $CH(P'^+)$ and $CH(P'^-)$, it is shown in

Theorem 4.1 that the set P is not hemispherical if and only if $CH(P'^{+})$ and $CH(P'^{-})$ intersects.

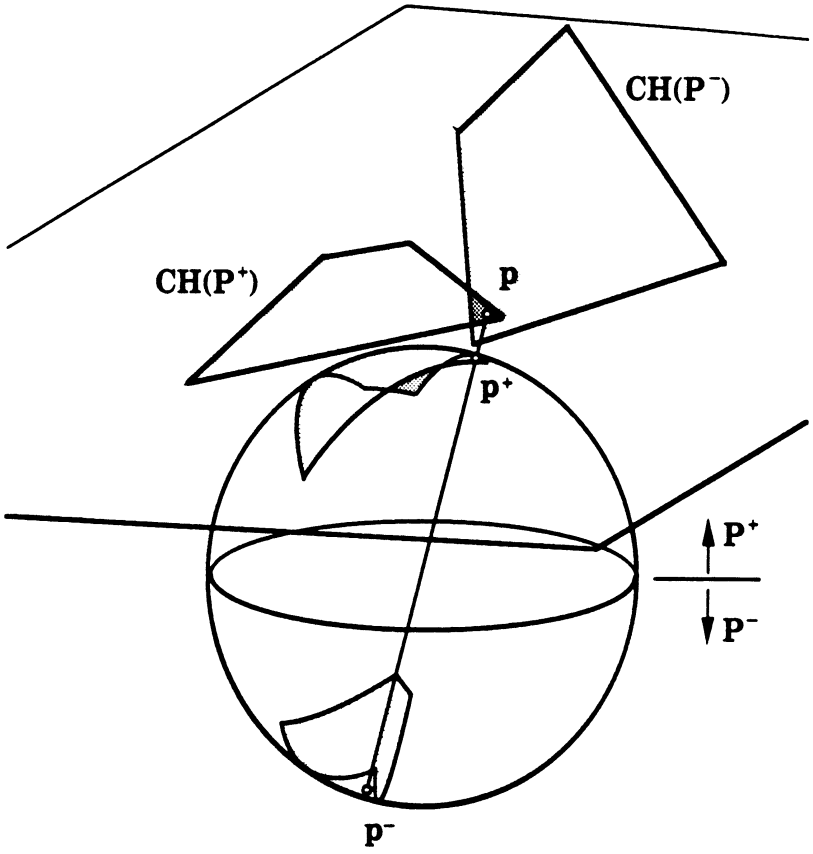
Theorem 4.1 *Given a set of N points P, P is not hemispherical if and only if* $CH(P'^{+})$ *and* $CH(P'^{-})$ *intersects, where CH(·) denotes the convex hull of a set of points in the euclidean plane and* P'^{+}, P'^{-} *denote the images of the points on the* $x_3 > 0$, $x_3 < 0$ *hemispheres, respectively, through central projection.*

Proof. First, it is shown that, if $CH(P'^{+})$ and $CH(P'^{-})$ intersect, P cannot be hemispherical. There exists a point $p' \in CH(P'^{+}) \cap CH(P'^{-})$ which is the image of a pair of antipodal points $p^{+} \in SCH(P^{+})$ and $p^{-} \in SCH(P^{-})$ as illustrated in Figure 4.1(a). Since $SCH(P^{+})$ and $SCH(P^{-})$ are the smallest convex sets containing P^{+} and P^{-}, respectively, any convex set containing P must contain both. Suppose H is a open hemisphere that contains P. Then H contains $SCH(P^{+})$ and $SCH(P^{-})$ which implies that it includes at lease a pair of antipodal points. This violates the definition of hemisphericity.

Second, it is shown that, if $CH(P'^{+})$ and $CH(P'^{-})$ do not intersect, P is hemispherical. For, if they do not, there exists a separating line SL such that $CH(P'^{+})$ and $CH(P'^{-})$ lie on opposite sides of it. Let the great circle corresponding to SL be L. Then S^2 is partitioned into 4 regions by L and the equator as shown in Figure 4.1(b). Suppose P^{-} exists in the quadrant (L^{-}, P^{-}). If P^{+} exists in the quadrant (L^{+}, P^{+}), then $CH(P'^{+})$ and $CH(P'^{-})$ are on the same side of SL by central projection. Since they are not, P^{+} can only exist in the quadrant (L^{-}, P^{+}). Hence, P must be hemispherical to the great circle L. Suppose P^{-} exists in the quadrant (L^{+}, P^{-}). Same arguments applies and P^{+} can only exist in the quadrant (L^{+}, P^{+}). Again, P must be hemispherical. ❑

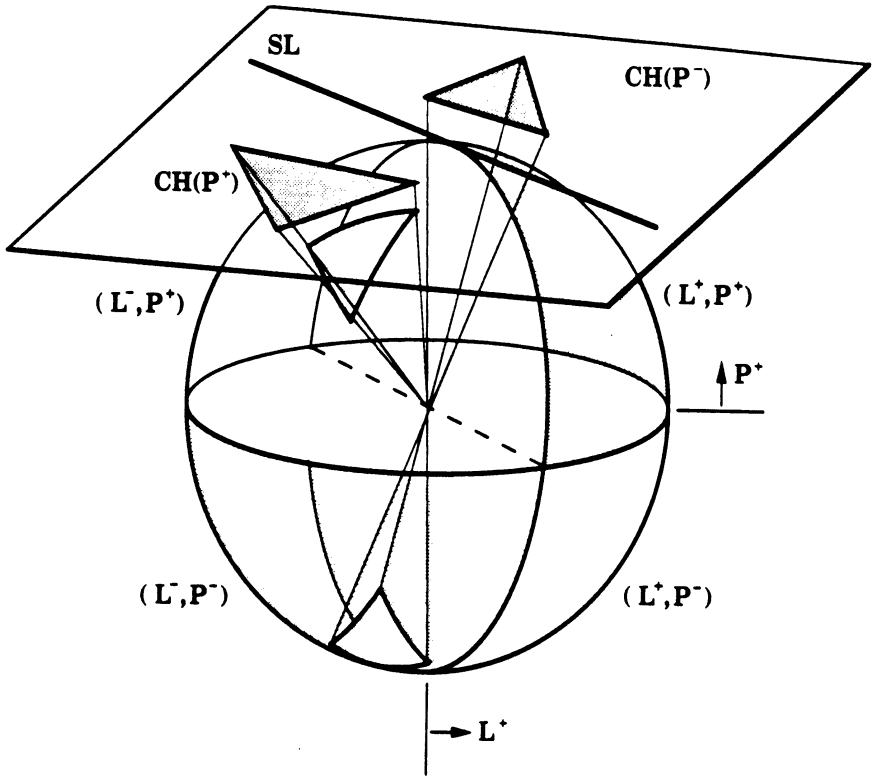
<Insert Figure 4.1>

The algorithm for detecting hemisphericity is immediate.



(a) $CH(P^+)$ and $CH(P^-)$ intersect

Figure 4.1 Hemisphericity



(b) $CH(P^+)$ and $CH(P^-)$ do not intersect

Algorithm S1: Hemisphericity

Step 1. Construct P'^{+} and P'^{-} by applying the central projection. If either set is empty, the set P is hemispherical. (This can be done in O(N) time.)

Step 2. Find $CH(P'^{+})$ and $CH(P'^{-})$. (This can be done in O(NlogN) time using Graham's algorithm [6] or other convex hull algorithms as described in [14].)

Step 3. Test if $CH(P'^{+})$ and $CH(P'^{-})$ intersects. (This can be done in O(N) time using the algorithm proposed by O'Rourke et al in [11].) If so, the set P is not hemispherical. otherwise, go to Step 4.

Step 4. P is hemispherical. Find a separating line for $CH(P'^{+})$ and $CH(P'^{-})$. (This can be done in O(N) time using Megiddo's algorithm as described in [9,16].) The plane contains the separating line and the origin determines the hemisphere.

Problem S2: Constructing Spherical Convex Hull. Given a set P of N points in S^2, construct its (spherical) convex hull, SCH(P).

By definition of convexity, if the set P is not hemispherical, SCH(P) is the entire sphere. Therefore, the first step is to detect whether P is hemispherical by applying Algorithm S1. Then, if P is hemispherical, SCH(P) can be constructed from the by-products of the hemisphericity detection algorithm – the two convex hulls, $CH(P'^{+})$ and $CH(P'^{-})$.

Algorithm S2: Spherical Convex Hull

Step 1. Determine if P is hemispherical using Algorithm S1. (This can be done in O(NlogN) time.) If P is not hemispherical, SCH(P) is the entire sphere; otherwise, go to Step 2.

Step 2. P is hemispherical. Construct SCH(P^+) from CH(P'^+) by connecting those points in S^2 that give rise to the vertices of CH(P'^+), in the order of tracing the boundary of CH(P'^+). (This can be done in O(N) time.) Similarly, construct SCH(P^-) from CH(P'^-).

Step 3. Let $L = \{ x \mid (a \cdot x) = 0, a = (a_1,a_2,a_3) \}$ be the great circle found at Step 1, which determines a hemisphere that completely contains P. Define a three dimensional rotation R which rotates $a = (a_1,a_2,a_3)$ to (0,0,1). (This can be done in constant time.) Then, after SCH(P^+) and SCH(P^-) are transformed by R, both are completely contained in S^{2+}, the hemisphere corresponds to $x_3 > 0$. (This can be done in O(N) time.)

Step 4. Construct CH(P''^+) and CH(P''^-) by applying central projection. (This requires O(N) time.)

Step 5. Find CH(P''), the convex hull of the union of CH(P''^+) and CH(P''^-). (This can be done in O(N) time using Shamos' algorithm described in [14] or Preparata-Hong's in [12].) Construct SCH(P) from CH(P'').

Figure 4.2 illustrates how SCH(P) can be constructed from the by-products of Algorithm S1. After SCH(P^+) and SCH(P^-) are transformed by the rotation defined in Step 3 of Algorithm S2, both are on the northern hemisphere (above the equator L). Through central projection, the two spherical convex hulls are mapped to two convex hulls, CH(P''^+) and CH(P''^-) in E^2. Then, the convex hull of the union of CH(P''^+) and CH(P''^-), the shaded region, is found by constructing the support lines. The complexity of this algorithm is O(NlogN), which is dominated by that of Step 1.

< Insert Figure 4.2>

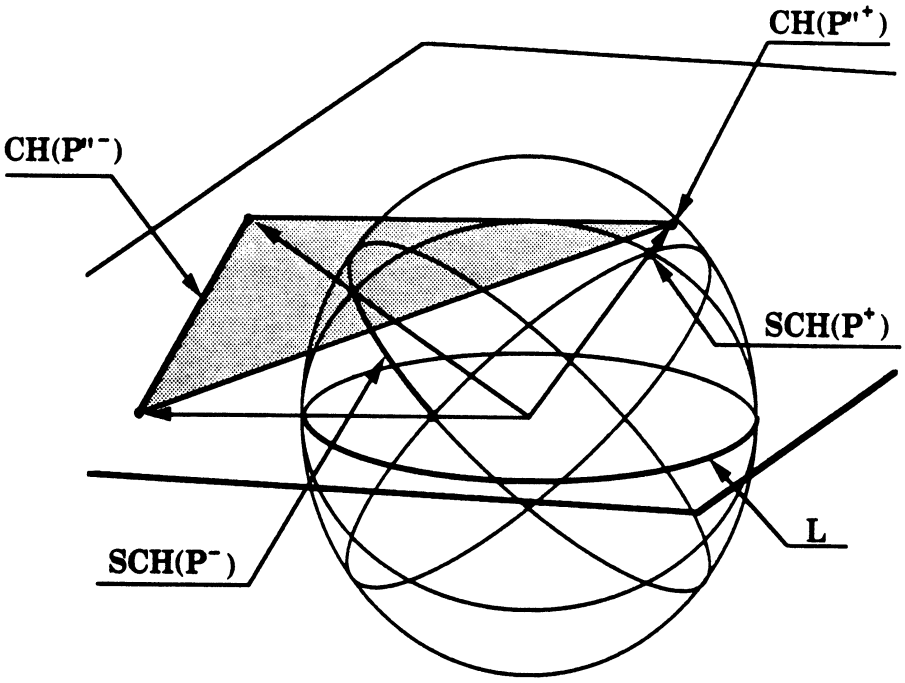


Figure 4.2 Construction of Spherical Convex Hull

Problem S3: Detecting Hemisphericity of a Union. Given a spherical convex polygon P with N vertices and a point p in S^2, determine if the union of P and p is hemispherical.

Problem S3 is fundamental, if a dynamic convex hull algorithm is required. If the union of P and p is found to be hemispherical, its convex hull can be constructed by using projection and planar algorithms; otherwise, its convex hull is the sphere. Without falling back to the O(NlogN) algorithm for Problem S1, S3 is converted to a convex polygon inclusion problem and a simple O(N) algorithm is presented.

> ***Algorithm S3*: Union Hemisphericity**
>
> Let the antipode of p be q. Determine whether q is internal to P. Since the (spherical) convex polygon P can be viewed as the intersection of the unit sphere S^2 and all the half-spaces, each of which is determined by an edge of P and the origin, q in S^2 is internal to P if it is internal to the intersection of all the half-spaces thus defined. Whether q lies in a half-space can be found by taking dot product of q and the normal of the plane defining the half-space. If q is internal to P, the union of P and p includes an antipodal pair, p and q, and, therefore, it is not hemispherical; otherwise, it is hemispherical.

This algorithm requires O(N) time because the detection of whether a point lies in a half-space takes constant time and the polygon has N edges.

Problem S4: Intersection of Hemispheres. Given N hemispheres H_1, H_2,...,H_N, find their intersection.

By observing the duality between a point and a hemisphere on the sphere, It is shown that Problem S4 is the dual of Problem S2.

Theorem 4.2 *Suppose a hemisphere is defined as* $H_i = \{ x \mid (p_i \cdot x) \geq 0, p_i = (p_{i1}, p_{i2}, p_{i3}), x \in S^2 \}$, $i = 1,...,N$. *Then, the intersection of hemispheres* $H_1 \cap H_2 \cap \cdots \cap H_N$ *exists if and only if the set P of points* p_i, $i = 1,...,N$, *is hemispherical.*

Proof. Suppose P is hemispherical. The intersection $H_1 \cap H_2 \cap \cdots \cap H_N$ exists since, for any j,k, $1 \leq j \leq N$, $1 \leq k \leq N$, $H_j \cap H_k$ is not empty. Suppose P is not hemispherical. Let $P^* = \{ p \mid (p \cdot q_j) > 0, p \in P \}$ where q_j is the antipode of some $p_j \in P$. Then P^* is the largest subset of P such that the union of P^* and q_j is hemispherical. Since P is not hemispherical, P^* is not empty. Let $I^* = \{ i \mid p_i \in P^* \}$. The intersection of H_i , $i \in I^*$, exists since P^* is hemispherical. Furthermore, this intersection lies within the open hemisphere $H_j' = \{ x \mid (p_j \cdot x) < 0, x \in S^2 \}$ which has no intersection with H_j. Therefore, the intersection $H_1 \cap H_2 \cap \cdots \cap H_N$ is empty. ❑

Theorem 4.3 *Let* $H_i = \{ x \mid (p_i \cdot x) \geq 0, p_i = (p_{i1}, p_{i2}, p_{i3}), x \in S^2 \}$, $i = 1,...,N$, *and P be the set of points* p_i, $i = 1,...,N$. *If P is hemispherical, each vertex of the spherical convex hull SCH(P) corresponds to an edge of the intersection* $H_1 \cap H_2 \cap \cdots \cap H_N$ *and vice versa.*

Proof. First, it is shown that the hemisphere determined by any point in SCH(P) that is not a vertex is redundant, i.e. it does not contribute to the boundary of the intersection $H_1 \cap H_2 \cap \cdots \cap H_N$. Let p_k be any point in the interior of the convex hull and $H_k = \{ x \mid (p_k \cdot x) \geq 0, x \in S^2 \}$ be the hemisphere defined by p_k. Suppose H_k contributes to an edge of the intersection. Let the edge be decided by the intersection of H_k and two other hemispheres, H_i and H_j. Since p_k is in the interior of SCH(P), there exists another point $p_m \in P$ in the neighborhood of p_k such that the distance between p_m and the edge contributed by H is greater than $\pi/2$. Then, the intersection of the hemisphere determined by p_m , H_i, and H_j is smaller than that of H_k, H_i, and H_j. Therefore, H_k does not contribute to the boundary. Now, let p_k be any point on an edge of SCH(P) and let p_i and p_j be the two end points of the edge. The intersection of the hemispheres

determined by p_k, p_i, and p_j has only two vertices (since they are collinear) and two edges (from p_i and p_j). Therefore, the hemisphere determined by p_k is redundant.

Second, it is shown that the hemisphere determined by a vertex of SCH(P) is non-redundant. Let p_k be a vertex of SCH(P) and H_k be the hemisphere determined by p_k. Suppose H_k is redundant. Then, there does not exist $x = (x_1, x_2, x_3)$ such that

$$p_{i1}x_1 + p_{i2}x_2 + p_{i3}x_3 \geq 0,\ 1 \leq i \leq N,\ i \neq k$$

$$p_{k1}x_1 + p_{k2}x_2 + p_{k3}x_3 < 0$$

By Farkas' Lemma [10], there exists some $y = (y_1, \ldots, y_{k-1}, y_{k+1}, \ldots, y_n)$ such that

$$p_{11}y_1 + \cdots + p_{k-1,1}y_{k-1} + p_{k+1,1}y_{k+1} + \cdots + p_{N1}y_N = p_{k1}$$

$$p_{12}y_1 + \cdots + p_{k-1,2}y_{k-1} + p_{k+1,2}y_{k+1} + \cdots + p_{N2}y_N = p_{k2}$$

$$p_{13}y_1 + \cdots + p_{k-1,3}y_{k-1} + p_{k+1,3}y_{k+1} + \cdots + p_{N3}y_N = p_{k3}$$

$$y_1, \ldots, y_{k-1}, y_{k+1}, \ldots, y_N \geq 0$$

Therefore, p_k can be written as a non-negative linear combination of p_i, $1 \leq i \leq N$, $i \neq k$. This contradicts the fact that p_k is a vertex. ❑

Figure 4.6 illustrates the duality between the convex hull of p_i and the intersection of H_i. Each edge of the intersection corresponds to a vertex of the convex hull. Based on Theorem 4.2 and Theorem 4.3, an O(NlogN) algorithm is proposed here for computing the intersection of hemispheres.

< Insert Figure 4.3>

Algorithm S4: Intersection of Hemispheres

Step 1. Determine if P is hemispherical. (This requires O(NlogN) time using Algorithm S1.) If P is not hemispherical, the intersection $H_1 \cap H_2 \cap \cdots \cap H_N$ is empty; otherwise, go to Step 2.

Step 2. Find SCH(P). (This can be done in O(N) time using Algorithm S2.)

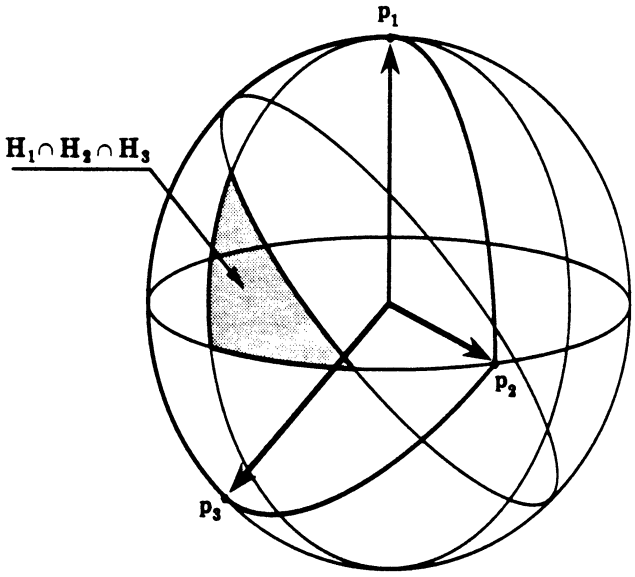


Figure 4.3 Duality between Convex Hull and Intersection

Step 3. Find a point p in the interior of SCH(P). By Theorem 4.3, this point determines a hemisphere that completely contains the intersection $H_1 \cap H_2 \cap \cdots \cap H_N$. (This can be done in O(N) time.)

Step 4. Traverse SCH(P) and construct the intersection $H_1 \cap H_2 \cap \cdots \cap H_N$. For each edge of SCH(P), the intersection of the two corresponding hemispheres determines a pair of antipodal points. One of the pair, which has a positive dot product with p (found at Step 3), is a vertex of $H_1 \cap H_2 \cap \cdots \cap H_N$. (This step can be done in O(N) time since the intersection of two hemispheres can be computed in constant time by taking the cross-product of the two unit normals.)

5. Conclusions

Unifying two fields of research – one that is analytically complex (but combinatorially simple) and the other combinatorially complex (but analytically simple) – may result in one of two outcomes. If an attempt had been made via the "lowest common denominator" of the two fields, e.g. the machining of planar surfaces, the outcome would have been predictably trivial. This paper reflects an attempt to combine the more appealing aspects of both fields by way of spherical geometry. To avoid duplication of and to give recognition to past work, central projection is adopted in this paper.

In computational efficiency, simplicity in structure offers many advantages. Convexity has been employed as the underlying theme in this paper in the form of hemisphericity and its related computations. Hemisphericity is justified by the practical application of NC machining, and , in particular, the geometry of the cutting tools and the degree of freedom in 3-, 4-, 5-axis milling machines.

Acknowledgement

The authors are grateful to the support of a grant from the Science Research Laboratory, Ford Motor Company. The authors also sincerely appreciates helpful insights and comments from their colleagues: J. Gan, D. S. Kim, S. Y. Chou, K. Tang, J. Y. Park, W. J. Lee, S. Y. Shin, J. Wolter, and D. Dutta.

References

[1] D. P. Anderson, Efficient algorithms for automatic viewer orientation, Comput. & Graphics , vol. 9, no. 4, 407-413 , 1985.

[2] K. Q. Brown, Geometric transformations for fast geometric algorithms, Ph. D Thesis, Dept. of Computer Science, Carnegie Mellon Univ., Dec. 1979.

[3] B. M. Chazelle, L. J. Guibas, and D. T. Lee, The power of geometric duality, BIT 25, 76-90, 1985.

[4] A. R. Forrest, Computational Geometry, Proc. Royal Society London, 321 Series 4, 187-195, 1971.

[5] T. Gonzalez, Algorithms on sets and related problems, Technical Report, Dept. of Computer Science, Univ. of Oklahoma, 1975.

[6] R. L. Graham, An efficient algorithm for determining the convex hull of a finite planar set, Info. Proc. Lett. 1, 132-133, 1972.

[7] D. Hilbert and S. Cohn-Vossen, Geometry and the Imagination, New York: Chelsea, 1952.

[8] B. K. Horn, Extended gaussian image, Proc. of the IEEE, vol. 72, no. 12, 1671-1686, Dec. 1984.

[9] N. Megiddo, Linear time algorithm for linear programming in R^3 and related problems, SIAM J. of Comput. 12(4), 759-776, Nov. 1983.

[10] K. G. Murty, Linear Programming, New York: Wiley, 1983.

[11] J. O'Rourke, C. Chien, T. Olson, and D. Naddor, A new linear algorithm for intersecting convex polygons, Computer Graphics and Image Processing 19, 384-391, 1982.

[12] F. P. Preparata and S. J. Hong, Convex hulls of finite sets of points in two and three dimensions, Comm. ACM 2(20), 87-93, Feb. 1977.

[13] F. P. Preparata and D. E. Muller, Finding the intersection of n half-spaces in time O(nlogn), Theoretical Computer Science 8(1), 45-55, Jan. 1979.

[14] F. P. Preparata and M. I. Shamos, Computational Geometry – An Introduction, Springer Verlag, 1985.

[15] P. J. Ryan, Euclidean and Non-Euclidean Geometry – An Analytic Approach, New York: Cambridge University Press, 1986.

[16] M. I. Shamos, Computational Geometry, Ph.D. Thesis, Dept. of Computer Science, Yale Univ., 1978.

[17] S. Y. Shin, Visibility in the Plane and Its Related Problems, Ph.D. Thesis, Dept. of Industrial and Operational Engineering, Univ. of Michigan, Ann Arbor, 1986.

[18] J. Stolfi, Primitives for Computational Geometry, Digital Systems Research Center Report 36, 1989.