SEPARATING AND INTERSECTING SPHERICAL POLYGONS FOR COMPUTING VISIBILITY ON 3-, 4-, AND 5-AXIS MACHINES

Lin-Lin Chen
Shuo-Yan Chou
and
Tony C. Woo
Department of
Industrial & Operations Engineering
University of Michigan
Ann Arbor, MI 48109-2117

Technical Report 91-9

March 1991

To appear in ASME Transactions, J. of Mechanical Design

SEPARATING AND INTERSECTING SPHERICAL POLYGONS FOR COMPUTING VISIBILITY ON 3-, 4-, AND 5-AXIS MACHINES

Lin-Lin Chen
Shuo-Yan Chou
and
Tony C. Woo
Department of
Industrial & Operations Engineering
University of Michigan
Ann Arbor, MI 48109-2117

Technical Report 91-9

March 1991

Separating and Intersecting Spherical Polygons: for Computing Visibility on 3-, 4-, and 5-axis Machines

LIN-LIN CHEN, SHUO-YAN CHOU, and TONY C. WOO
Department of Industrial and Operations Engineering, University of Michigan

Abstract. For the determination of optimal workpiece orientations for 3-, 4-, and 5-axis numerical control machining and coordinate measurement, sculptured surfaces are mapped on to the Gaussian sphere. Based on the cutting tool geometry, the Gaussian maps are transformed into visibility maps which are spherical polygons.

Given a set of n spherical polygons $\mathcal{P}$, we find a hemisphere that contains the largest number of polygons in $\mathcal{P}$ (for 3-axis machines), a great circle that divides $\mathcal{P}$ as equally as possible (for 3-axis machines with two setups), and a great circle that cuts the largest (or the smallest) number of polygons in $\mathcal{P}$ (for 4-axis machines), from which 5-axis machining is then shown to be a simple extension. These are enabled by an O(vn log n) time algorithm which computes all the possible ways of cutting $\mathcal{P}$ as a partitioning of the unit sphere induced by 2n spherically-convex polygons, where v is the total number of vertices with each face of the partitions representing a possible cut of $\mathcal{P}$, all cuts can be enumerated in O(nv) time.

CR Categories and Subject Descriptions: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling–*geometric algorithms, languages and systems*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems–*geometrical problems and computations;* J.6 [**Computer-Aided Engineering**]: Computer-aided manufacturing (CAM)

General Terms: Algorithms, Design, Performance, Manufacturing

Additional Key Words and Phrases: Spherical Algorithms, visibility Algorithms, separation, bisection, minimal/maximal intersection, densest hemisphere, numerical control machining, coordinate measurement

1. INTRODUCTION

While tool paths can be computed readily from the computer representation of sculptured surfaces, their validity depends on the type of machine and the orientation of the workpiece. In this paper, we consider the extrinsic constraints (the number of axes in a milling machine or a coordinate measurement machine and the number of setups for a workpiece) together with the intrinsic geometry of the sculptured surfaces, with two objectives in mind: machine selection (as minimization of the number of axes) and workpiece orientation (as minimization of the number of setups).

The normals of a surface are often used to orient a tool. Figure 1 shows that this convention is unnecessary. In Figure 1.1(a), the axis of a flat-end milling cutter is aligned with the normal at a point in a surface to avoid excessive gouging. The use of a ball-end milling cutter is illustrated in Figure 1.1(b). As the tip of the ball-end mill has zero radial velocity, material removal is more efficient if the cutter axis is *not* aligned with a surface normal, as illustrated in Figure 1.1(c). The same can be said for the probe of a coordinate measurement machine.

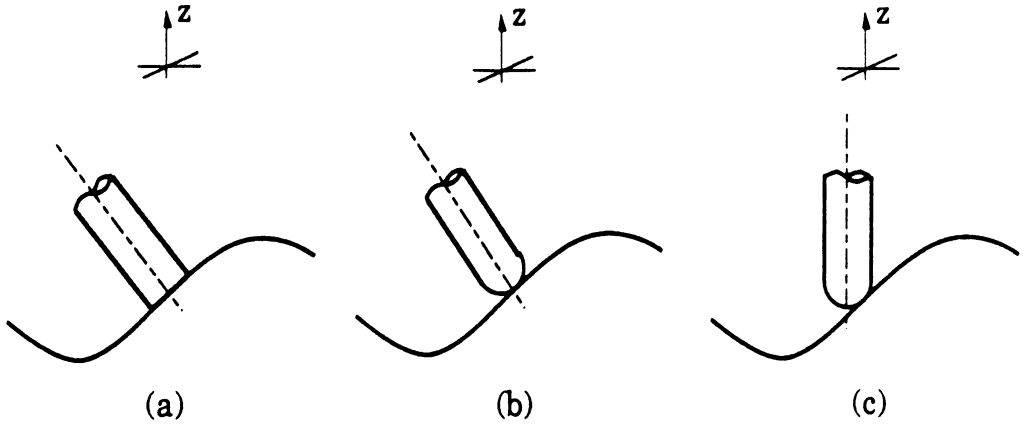

Figure 1.1 Flat-end and ball-end tools

The observation illustrated by Figure 1 leads to the notion of *visibility* (or accessibility) by a tool: the visibility of a point can be enhanced by up to 180°, if a ball-end tool is used. The visibility of (all the points in) a surface (to a ball-end tool) can then be computed readily. Given a surface, its normals form a *Gaussian map* (or G-Map) on a unit sphere [12], i.e., a point in a G-Map corresponds to a surface normal. Enhancing the visibility of a single point in a G-Map by 180° gives a hemisphere, as illustrated in Figure 1.2(a). For all points in the G-Map to be simultaneous visible to a ball-end tool, their corresponding hemispheres are intersected as shown in the sequence in Figures 1.2(b) and 1.2(c). We call such an intersection of hemispheres the *visibility map* (or V-Map) of a sculptured surface. In other words, the workpiece should be oriented such that the V-Map of the surface contains the given tool axis.

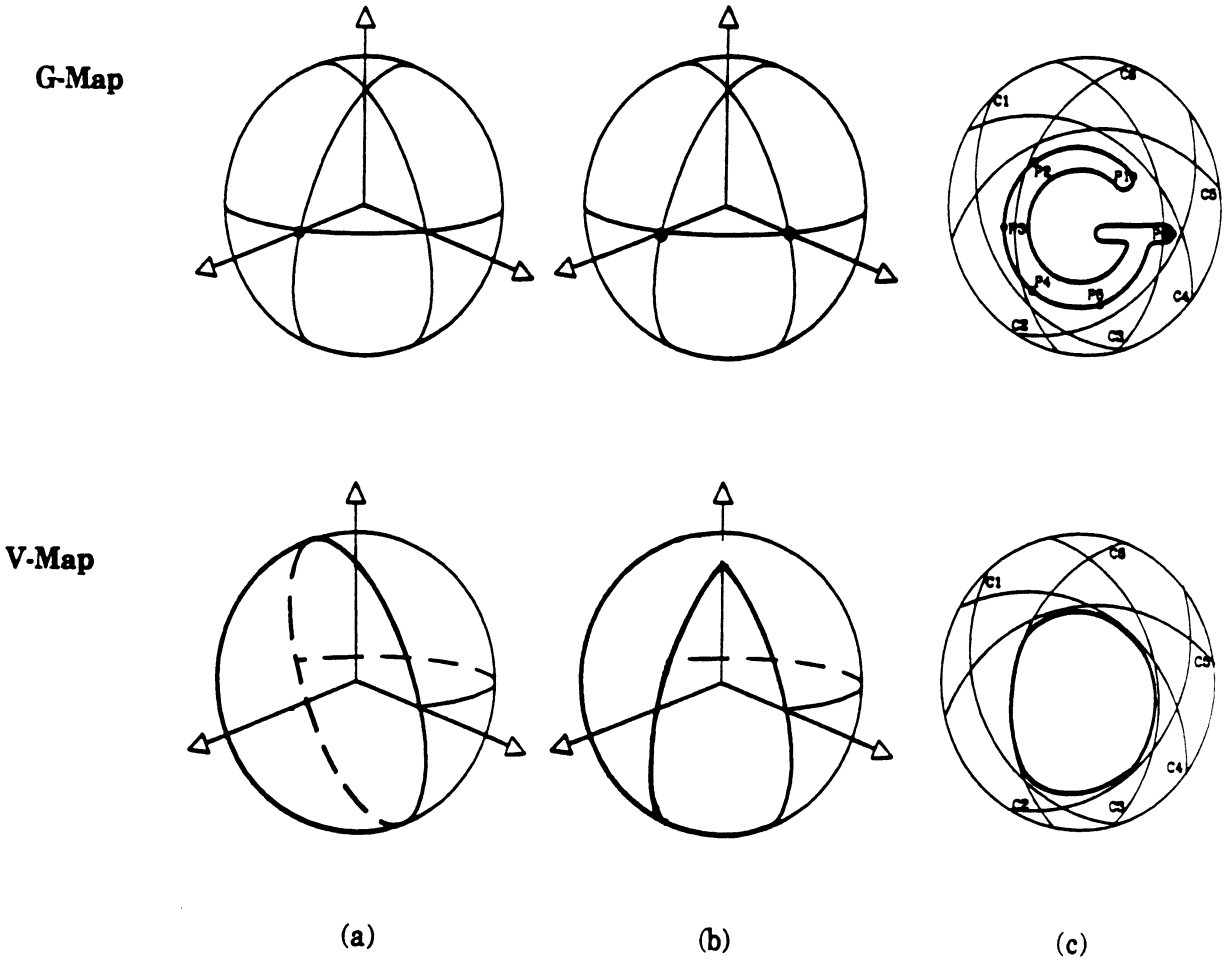

Figure 1.2 Gaussian and visibility maps

Thus, a mechanical object with n surfaces can be represented by n spherical polygons. If the spherical polygons are V-Maps, and if k of the n spherical polygons intersect jointly, the corresponding k surfaces can be machined in a single setup. If, on the other hand, the

spherical polygons are the (unenhanced) G-Maps, they must all be contained in a hemisphere for them to be machined in a single setup.

Implicit to the above discussion is 3-axis machining, the least expensive and the most available amongst the 3-, 4-, and 5-axis numerically controlled machines. Revisiting Figure 1.1 confirms the (hidden) assumption that the tool axis of a 3-axis machine is fixed. However, machining with a 3-axis could also imply more setups (dismounting, re-orienting, and then re-clamping the workpiece) than does on a 4- or 5-axis. This intuition is validated by Figure 1.3, which shows that the worktable has one and two more degrees of freedom, for a 4- and a 5-axis machines, respectively, than does on a 3-axis machine. If the worktable of a 4-axis rotates 360°, the cutter traces a great circle on the Gaussian sphere. To exploit this extra degree of freedom so that the machine "sees" as many surfaces in a single setup as possible, we seek the orientation of a great circle such that it intersects (or cuts) the V-Maps maximally. The fifth axis typically as a limited rotation (of 30° – 80°, depending on the manufacturer). Workpiece orientation on a 5-axis machine therefore corresponds to the determination of maximal intersection between the VMaps and a spherical band.

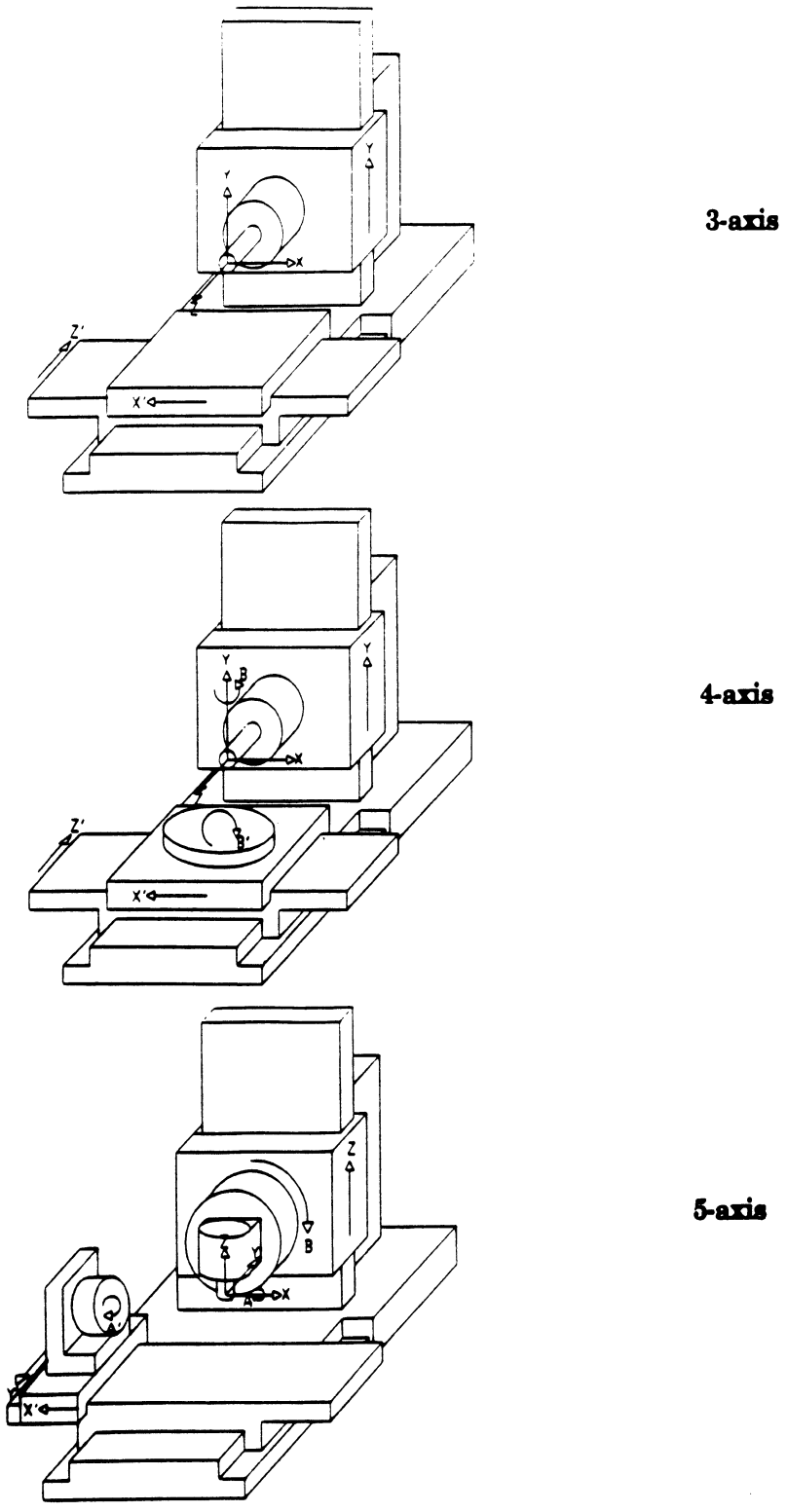

Figure 1.3 3-, 4-, and 5-axis numerically controlled machines

We are now ready to formulate the problem of workpiece orientation on the sphere. We assume that a variety of 3-, 4-, and 5- axis machines with ball-end tools are available; and that a mechanical component with n surfaces is represented by n spherical polygons.

Densest Hemisphere: Given a set of n spherical polygons that are G-Maps, find a hemisphere that contains the largest number of them. (This is the "greedy" algorithm for selecting a 3-axis machine and for determining the workpiece orientation.)

Bisector: Given a set of n G-Maps, find a great circle that separates them as equally as possible. (This formulation limits the number of setups on a 3-axis machine to two, though it is not always possible to access all the given surfaces.)

Maximal Intersection: Given a set of n V-Maps (that are enhanced G-Maps), find a great circle that intersect the maximal number of them. (This corresponds to the "greedy" algorithm for selecting a 4-axis machine and for determining the workpiece orientation.)

In this paper, we report algorithms for solving these problems along with some variations. As a bisection is a special case of a separator, we find all the separators for a given set of spherical polygons.

Separator: Given a set of n G-Maps, find a great circle that separates them.

As finding the minimal intersection is an extension to finding no intersection (for a separator) and it incurs no additional cost to finding maximal intersection, we report them as well.

Minimal Intersection: Given a set of n V-Maps, find a great circle that intersect the minimal number of them.

Together, these five problems are solved in one-fell-swoop by first computing a partitioning of the sphere by 2n *convex* polygons based on the n given polygons in O(vn log n) time, where v is the total number vertices of the n polygons, and the partitions enumerated in O(nv) time. In the course of development, some of the allied computational supports have been previously reported [6]: detecting the hemisphericity (of a G-Map), intersecting n hemispheres (of n points in a G-Map to construct a V-Map), and determining the spherically convex hull (of a G-Map as an approximation).

2. DUALITY ON THE SPHERE

Much like the duality between a point and a line in the Euclidean plane [4,5], there is a duality between a point and a hemisphere on the unit sphere. By utilizing this duality, the problems of finding certain great circles can be converted to the problems of finding certain points. Thus, we can transform the problem of finding bisectors to that of finding sets of maximally covered points, and at the same time, equate the problem of finding maximally intersecting great circles to that of finding sets of minimally covered points. To establish spherical duality as the basis of our algorithms, we first review elements from spherical geometry [19].

Let E^3 denote the three-dimensional Euclidean space and S^2 denote the (two-dimensional) surface of the unit sphere. Then, $S^2 = \{ p : |p| = 1, p \in E^3 \}$. A *point* p in S^2 is a unit vector in E^3 and is represented by a 3-tuple (x_1,x_2,x_3). Two points p and q are *antipodal* if $p = -q$, where p is the *antipode* of q and vice versa. A *line* in S^2 is a *great circle,* the intersection of the sphere with a plane containing the center of the sphere, the origin. (We shall use the term *line* and the term *great circle* interchangeably.) The surface of the sphere is divided into two *hemispheres* by a great circle. A *closed hemisphere* includes the bounding great circle, while an *open hemisphere* does not.

For any two non-antipodal points p and q in S^2, the *line segment* $\bar{pq}$ joining them is the shorter of the two arcs in the great circle con-

taining p and q. If p and q are antipodes, then any great semicircle joining p and q is a segment. The *distance* between p and q is defined by the metric: $d(p,q) = \cos^{-1}(p \cdot q)$.

A *polygon* P on the sphere is a closed path of ordered line segments or edges $\bar{p_1p_2}, \bar{p_2p_3}, \cdots, \bar{p_{n-1}p_n}, \bar{p_np_1}$ that admits no self-intersection; the points $p_1, p_2, \cdots, p_n$ are the *vertices* of P. The *opposite* of a spherical polygon P is another spherical polygon ~P represented by ordered edges $\bar{q_1q_2}, \bar{q_2q_3}, \cdots, \bar{q_nq_1}$ where q_i is the antipode of p_i.

A set of points P in S^2 is *quasi-convex* if, for any pair of non-antipodal points p and q in P, the segment $\bar{pq}$ is also in P. (Note that the only quasi-convex set that is not contained in any closed hemisphere is the entire sphere.) A *spherically-convex hull* of a set of points P in S^2, denoted by SCH(P), is the smallest (quasi-) convex set in S^2 containing P.

Using these terminologies, we now define the duality between a point and a hemisphere:

Definition. *The dual of a point p on the sphere is a (closed) hemisphere* $H = \{ x \mid (p \cdot x) \geq 0, x \in S^2 \}$ *and vice versa.*

Through the point-hemisphere duality, we establish the duality between two special kinds of convex polygons: one formed by taking the spherically-convex hull of a set of points and the other by intersecting the dual hemispheres of the points in the set.

Theorem 2.1 *Let P be a set of points p_i, $i = 1, \cdots, n$, and H_i be the dual hemisphere of p_i. If P is not hemispherical (i.e. is not contained in some closed hemisphere), then $H_1 \cap H_2 \cap \cdots \cap H_n = \emptyset$. Otherwise, each vertex of SCH(P), the spherically-convex hull of P, corresponds to an edge of the intersection $H_1 \cap H_2 \cap \cdots \cap H_n$ and vice versa.*

Proof. We first show that, if P is not hemispherical, then the hemispheres $H_1, H_2, \cdots, H_n$ have no common intersection. Assume that P is not entirely contained in any closed hemisphere. Then, the set $P^* = \{ x \mid (p_i \cdot x) \geq 0, \ i = 1,..., n, \ x \in S^2 \}$ must be empty. But P^* is also the common intersection of hemispheres $H_1, H_2, \cdots, H_n$. Thus, $H_1, H_2, \cdots, H_n$ must not have a common intersection.

Next, if P is hemispherical, then each vertex of SCH(P) corresponds to an edge of the intersection $H_1 \cap H_2 \cap \cdots \cap H_n$ and vice versa. Let $p_j \in P$. We show that H_j is not redundant (i.e. it contributes to the boundary of the intersection $H_1 \cap H_2 \cap \cdots \cap H_n$) if and only if p_j is a vertex of SCH(P).

If p_j is a vertex of SCH(P), it must lie outside of SCH(P – $\{p_j\}$), the spherically-convex hull of the point set P – $\{p_j\}$. Therefore, $p_j = (p_{j1}, p_{j2}, p_{j3})$ cannot be expressed as a positive combination of the other vectors in P, i.e., there does not exist a vector $y = (y_1,...,y_{j-1},y_{j+1},...,y_n)$ such that

$$p_{11}y_1 + \cdots + p_{j-1,1}y_{j-1} + p_{j+1,1}y_{j+1} + \cdots + p_{n1}y_n = p_{j1}$$

$$p_{12}y_1 + \cdots + p_{j-1,2}y_{j-1} + p_{j+1,2}y_{j+1} + \cdots + p_{n2}y_n = p_{j2}$$

$$p_{13}y_1 + \cdots + p_{j-1,3}y_{j-1} + p_{j+1,3}y_{j+1} + \cdots + p_{n3}y_n = p_{j3}$$

$$y_1, \cdots, y_{j-1}, y_{j+1}, \cdots, y_n \geq 0.$$

Then, by Farkas' Lemma [16], there exists some $x = (x_1, x_2, x_3)$ such that

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

$$p_{j1}x_1 + p_{j2}x_2 + p_{j3}x_3 < 0\ .$$

This means that the intersection of $H_1 \cap \cdots \cap H_{j-1} \cap H_{j+1} \cap \cdots \cap H_n$ and the complement of H_j is not empty. Consequently, H_j cannot be redundant.

If p_j is not a vertex of SCH(P), then p_j can be expressed as a positive combination of the other vectors in P. Again, by Farkas' Lemma, the intersection of $H_1 \cap \cdots \cap H_{j-1} \cap H_{j+1} \cap \cdots \cap H_n$ and the complement of H_j must be empty. Thus, H_j completely contain $H_1 \cap \cdots \cap H_{j-1} \cap H_{j+1} \cap \cdots \cap H_n$. Conversely, H_j must be redundant. ❑

It follows directly from Theorem 2.1 that, given a convex polygon P, the dual of P is the intersection of the dual hemispheres of the vertices of P. In the following, we will use P* to denote the dual of a convex polygon P; the term "convex" is understood to mean "spherically convex".

Using the point-hemisphere duality, we now describe how the relationship between a great circle and a convex polygon P can be characterized by way of the relationship between a point and two convex

polygons, P* and ~P*, the dual of P and the opposite of P*, respectively. Here and throughout, $i(\cdot)$ and ß $(\cdot)$ denote the interior and the boundary, respectively, of a set.

Lemma 2.2 *Let P be a convex polygon. For a given point q on the sphere, let*

$$H^+ = \{ x \mid (q \cdot x) \geq 0, x \in S^2 \} \text{ and}$$

$$H^- = \{ x \mid (q \cdot x) \leq 0, x \in S^2 \}.$$

Then,

(1) $P \subset i(H^+) \Leftrightarrow q \in i(P^*)$;

(2) $P \subset i(H^-) \Leftrightarrow q \in i(\sim P^*)$;

(3) $P \subset H^+$ and $P \cap \beta(H^+) \neq \emptyset \Leftrightarrow q \in \beta(P^*)$;

(4) $P \subset H^-$ and $P \cap \beta(H^-) \neq \emptyset \Leftrightarrow q \in \beta(\sim P^*)$; and

(5) $P \cap i(H^+) \neq \emptyset$ and $P \cap i(H^-) \neq \emptyset \Leftrightarrow q \notin (P^* \cup \sim P^*)$.

Proof. Trivial. ❑

As a great circle is the boundary of a hemisphere, we immediately have a corollary stating that the bounding great circle β(H) of a hemisphere H has no intersection with a convex polygon P if and only if the dual point q of H lies in the interior of P* or in the interior of ~P*; β(H) supports P if and only if q lies on the boundary of P* or ~P*; and β(H) intersects the interior of P if and only if q lies outside of P* and ~P*, as illustrated in Figure 2.1.

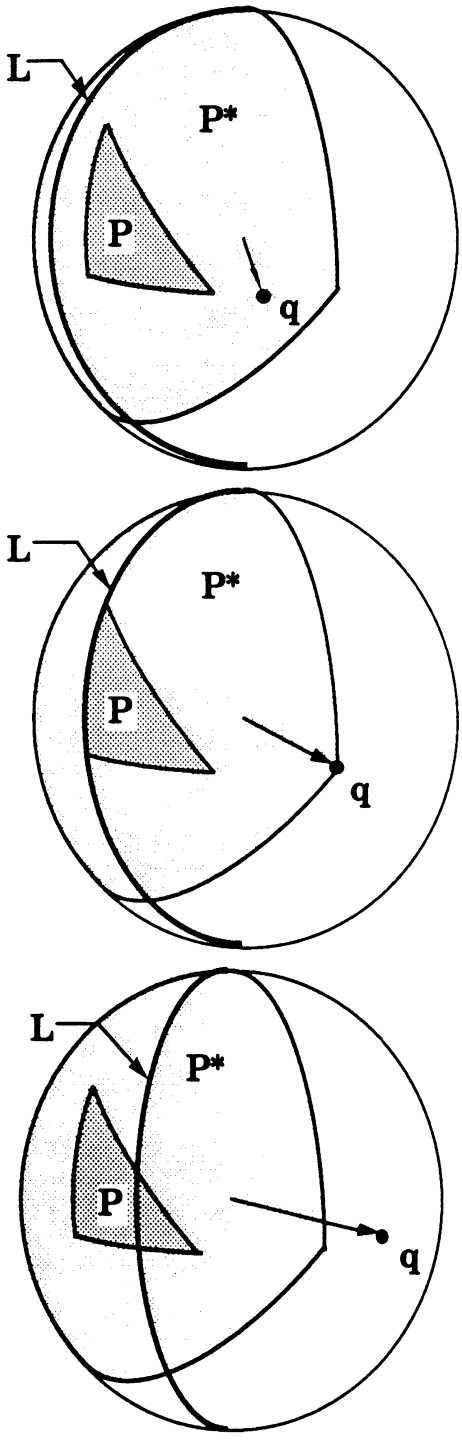

Figure 2.1 Relationships between L and P are characterized by relationships between q and P*.

Corollary 2.3 *Let P be a convex polygon on the sphere. For any great circle $L = \{ x \mid (q \cdot x) = 0, x \in S^2 \}$,*

(1) $L \cap P = \emptyset \Leftrightarrow q \in i(P^*)$ or $q \in i(\sim P^*)$,

(2) $L \cap i(P) = \emptyset$ and $L \cap P \neq \emptyset \Leftrightarrow q \in \beta(P^*)$ or $q \in \beta(\sim P^*)$, and

(3) $L \cap i(P) \neq \emptyset \Leftrightarrow q \notin (P^* \cup \sim P^*)$.

By observing that a great circle intersects a polygon if and only if it intersects its convex hull, we can extend the duality to any spherical polygon, convex or otherwise:

Theorem 2.4 *For any great circle $L = \{ x \mid (q \cdot x) = 0, x \in S^2 \}$, if a polygon P is not hemispherical, then $L \cap P \neq \emptyset$;*

otherwise,

(1) $L \cap P = \emptyset \Leftrightarrow q \in i(SCH(P)^*)$ or $q \in i(\sim SCH(P)^*)$;

(2) $L \cap i(P) = \emptyset$ and $L \cap P \neq \emptyset$

$\Leftrightarrow q \in \beta(SCH(P)^*)$ or $q \in \beta(\sim SCH(P)^*)$; and

(3) $L \cap i(P) \neq \emptyset \Leftrightarrow q \notin (SCH(P)^* \cup \sim SCH(P)^*)$.

Proof. If P is not hemispherical, then by definition the intersection of L and P is not empty. For the second part of the theorem, we observe the following:

(a) $L \cap P = \emptyset \Leftrightarrow L \cap SCH(P) = \emptyset$.

(b) $L \cap i(P) = \emptyset$ and $L \cap P \neq \emptyset$

$\Leftrightarrow L \cap i(SCH(P)) = \emptyset$ and $L \cap SCH(P) \neq \emptyset$.

(c) $L \cap i(P) \neq \emptyset \Leftrightarrow L \cap i(SCH(P)) \neq \emptyset$.

Once we have (a), (b), and (c), the second part of the theorem follows immediately from Corollary 2.3. ❑

We have classified great circles with respect to a single polygon by utilizing the point-hemisphere duality. In the next section, we will examine the classification of great circles when more than one spherical polygon is given.

3. Classification of Equivalent Great Circles

Given a set of spherical polygons $\mathcal{P} = \{ P_i \mid i = 1,..., n \}$ that are convex or otherwise, we classify great circles based on their relationships with the polygons in $\mathcal{P}$. Let us define two great circles L_1 and L_2 to be *cut-equivalent* (or equivalent, for short), if they intersect the same subset of polygons. (To simplify the discussion, if a great circle supports a polygon, we consider it intersecting the polygon.) By invoking the point-hemisphere duality just established, we show that the dual points of a set of equivalent great circles fall inside specific faces of the partitioning of the sphere induced by the convex polygons, $SCH(P_i)^*$ and $\sim SCH(P_i)^*$, i = 1,..., n.

Corollary 3.1. *Let P_i, $i = 1,..., n$, be polygons that are individually hemispherical. Then, a great circle L intersects only polygons P_i, $i = 1,..., m$, if and only if the dual point q of L satisfies the following:*

$$\begin{cases} q \notin i(SCH(P_i)^*) \text{ and } q \notin i(\sim SCH(P_i)^*), & \text{for } i = 1, ..., m, \text{ and} \\ q \in i(SCH(P_i)^*) \text{ or } q \in i(\sim SCH(P_i)^*), & \text{for } i = m+1, ..., n. \end{cases}$$

Proof. Follows directly from Theorem 2.4. ❑

In Corollary 3.1, it is necessary that all the n polygons be individually hemispherical. Otherwise, SCH(P) would be the entire sphere and SCH(P)* would be the empty set. This limitation, however, is not severe, since, if a polygon is not hemispherical, it intersects all great circles. Thus, algorithmically, non-hemispherical polygons are easy to handle; for each great circle, we simply increase the number of intersections by the number of non-hemispherical polygons.

From Corollary 3.1, by using the partitions on the sphere induced by the polygons, SCH $(P_i)^*$ and $\sim SCH(P_i)^*$, i = 1,..., n, if two points p and q fall inside the same partition (called a *face*), then we know that the great circles determined by p and q intersect the same set of polygons. (Note that the converse is not true.) The great circles determined by p and q are thus equivalent. This implies that the number of sets of equivalent great circles cannot be greater than the number of faces in the partitions. Therefore, once the partitions are available, it is possible to enumerate all sets of equivalent great circles by

traversing the faces. Our algorithm for computing all sets of equivalent great circles follows:

Algorithm COMPUTE-ALL-SETS-OF-EQUIVALENT-GREAT-CIRCLES

Input: A set of spherical polygons $\mathcal{P} = \{ P_i \mid i = 1,..., n \}$.

Output: Partitioning of the sphere. In addition, implicitly associated with each face of the partitions is an ownership vector denoting the subset of polygons intersected by the dual line of a point in the face.

Step 1: For each spherical polygon in $\mathcal{P}$, determine whether it is hemispherical. Let $\mathcal{P}_H$ denote the subsets of hemispherical polygons.

Step 2: For each polygon P in $\mathcal{P}_H$, compute its spherically-convex hull, SCH(P), and obtain its dual, SCH (P)*. Let $\mathcal{C} = \{ SCH(P)^*, \sim SCH(P)^* \mid P \in \mathcal{P}_H \}$.

Step 3: Compute the partitions of the sphere induced by the convex polygons in $\mathcal{C}$.

Step 4: For each face of the partitions, record (implicitly) the ownership vector of the face.

Let v be the total number of vertices of polygons in $\mathcal{P}$. For Step 1, we can determine $\mathcal{P}_H$ in O(v) time by formulating the hemisphericity detection problem as a linear program and applying algorithms described in [7,15]. We can then compute for Step 2 the convex hull for

each polygon in $\mathcal{P}_H$ using O(v) time, taking advantage of the fact that we are given polygons as opposed to a point set. Once we have the convex hulls, we can obtain $\mathcal{C}$ in $O(v_H)$ time, where v_H is the number of vertices of the convex hulls, by applying the algorithm described in [6]. (In the worst case, $v_H = v$.)

For Step 3, we construct the partitions induced by the set of convex polygons $\mathcal{C} = \{C_1,..., C_n, \sim C_1,..., \sim C_n\}$, where $C_i = SCH(P_i)^*$. To each point p on the sphere, we assign an *ownership vector* $u(p) = (u_1(p), u_2(p),...,u_n(p))$, where

$$u_i(p) = \begin{cases} 1, & \text{if } p \in i(C_i); \\ -1, & \text{if } p \in i(\sim C_i); \\ 0, & \text{otherwise.} \end{cases}$$

(Note that, since $i(C_i)$ and $i(\sim C_i)$ have no intersection, it is impossible to have $p \in i(C_i)$ and $p \in i(\sim C_i)$.) Let L be the dual great circle of p. Then, from Corollary 3.1, the ownership vector has

$$u_i(p) \begin{cases} = 0, & \text{if L intersects the polygon that corresponds to } C_i; \\ \neq 0, & \text{otherwise.} \end{cases}$$

Following the definition of equivalence of two great circle L_1 and L_2, let two points p and q be *equivalent* if u(p) = u(q). We then define a *face* of the partitions to be the largest connected subset of equivalent points. We then let the *ownership vector* of a face F be the same as that of any point in F, i.e., u(F) = u(p), for any $p \in F$. Figure 3.1 illustrates an example of a face and its ownership vector. In the Appendix, we present an O(nv log n) time algorithm for constructing this spherical partitioning.

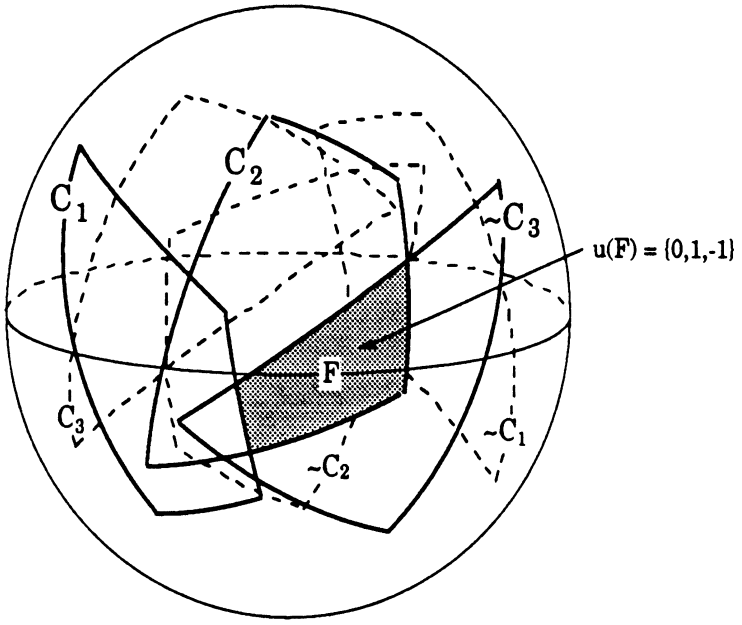

Figure 3.1 A face in a partitioning and its ownership vector

At Step 4, we record implicitly the ownership vector for each face in the partitions. To start with, each face can be covered by the convex polygons contributing to its boundary. However, simply accounting for these polygons is not sufficient, because a face can also be covered by polygons completely containing the face. Now, computing the ownership vector for each face from scratch would require $O(n^2 v \log(\frac{v}{n}))$ time because there are O(nv) faces and $O(n \log(\frac{v}{n}))$ time is required to determine the ownership vector of a face†. To re-

† If a polygon C_i contributes to the boundary of a face, then we can determine whether C_i covers the face in constant time; otherwise, we can do so in $O(\log v_i)$

duce this time complexity, we make the following observation: The two sets of convex polygons covering a pair of adjacent faces differ by exactly one element – the convex polygon whose edges separate the two faces. By utilizing the adjacency relationship among the faces, we can obtain the ownership vectors through propagation. The time complexity becomes $O(n \log (\frac{v}{n}) + nv (T_P))$, where T_P is the time to propagate the ownership vector from a face to an adjacent one. Here, if we explicitly store the ownership vector of each face, then $T_P \geq O(n)$ (because making a copy of an ownership vector takes O(n) time) and the time complexity of propagation is at least $O(n^2 v)$. We can further reduce the time complexity by storing the ownership vectors implicitly: We store the ownership vector of an arbitrary face along with a face-adjacency graph recording the difference between the ownership vectors of two adjacent faces. Then, $T_P = O(1)$ and we can complete the propagation in O(nv) time.

Since we can compute the ownership vector of an arbitrary face trivially, we only describe how we can construct the face-adjacency graph. Let G_F denote the face-adjacency graph where each node represents a face and two nodes are connected with an edge if their corresponding faces are adjacent. For each node, we store the face-id; for each arc, we store the polygon-id that is different between the

---

time. Thus, we can compute the ownership vector of a face in $\sum_{i=1}^{n} O(\log v_i) \leq O(n \log (\frac{v}{n}))$ time.

ownership vectors of the two faces corresponding to the end points of the edge. Then, G_F is the dual graph of the spherical partitioning obtained in Step 3. We can therefore construct the face-adjacency graph by performing a breath-first search on the faces, which requires O(nv) time. Consequently, we can implicitly represent the ownership vectors for all faces in O(nv) time.

In summary, we have presented an algorithm that computes a representation of all sets of equivalent great circles with respect to a given set of polygons. For each set of equivalent great circles, this representation also records the subset of polygons intersected by the great circles. The time complexity of this algorithm is dominated by Step 3 which requires O(nv log n) time.

4. APPLICATIONS

By using the dual representation of all sets of equivalent great circles, many problems involving the finding of certain great circles can be solved efficiently. We illustrate with five examples.

4.1. Separation

Given a set $\mathcal{P}$ of n polygons on the sphere, any great circle that does not intersect any of the polygons in $\mathcal{P}$ divides the set into two subsets,

$\mathcal{P}^+$ and $\mathcal{P}^-$, one of which may be empty. We call such a great circle a *separator* of the set $\mathcal{P}$.

Problem P.1 (All Separators). Given a set $\mathcal{P}$ of n polygons on the sphere, find all separators of $\mathcal{P}$.

To find all separators, we first apply Algorithm Compute-Sets-of-Equivalent-Great-Circles to obtain a representation of all equivalent great circles. Since equivalent great circles intersect the same subset of polygons in $\mathcal{P}$, if a great circle L is a separator of $\mathcal{P}$, then all great circles equivalent to L are also separators. What remains is to determine which sets of the equivalent great circles are separators. Since a separator admits no intersection with the polygons in $\mathcal{P}$, the ownership vector of its dual point, u(q), consists of either 1's or -1's. (Recall that if $u_i(q) = 0$ then the dual great circle of q intersects polygon $P_i \in \mathcal{P}$.) Thus, we have the following algorithm.

Algorithm FIND-ALL-SEPARATORS

Step 1. Apply Algorithm Compute-All-Sets-of-Equivalent-Great-Circles.

Step 2. Find all set of equivalent great circles with ownership vector consists of either 1's or -1's.

Traverse all sets of equivalent great circles starting from the face whose ownership vector is explicitly stored.

Count the number of zero terms, k, in this ownership vector. If $k = 0$, then the corresponding set of equivalent great circles are separators. The ownership vector and k are updated during the traversal in constant time with the aid of the face-adjacency graph.

Since Step 1 requires O(nv log n) time and Step 2 requires O(nv) time, the computing of the representation of all separators, can be achieved in O(nv log n) time† .

4.2. Bisection

As in the applications of 3-axis machining and coordinate measurement, it may be desirable to find separators which divides a given set of n polygons equally. Given a set $\mathcal{P}$ of n polygons on the sphere, a

---

† We note here that a representation of the separators of $\mathcal{P}$ can also be obtained using the following algorithm: Let $\mathcal{P} = \{ P_i \mid i = 1, ..., n \}$. Then, we can construct the dual polygons $\{ SCH (P_i)^* \mid i = 1, ..., n \}$ in O(v) time. We can then centrally project these dual polygons onto the $x_3 = 1$ plane. It is not difficult to show that the dual points of the separators fall inside the common intersection of the projected polygons. Since computing this intersection requires O(v log v) time, a representation of the separators can be obtained in O(v log v) time. This approach, however, does not apply when we need to distinguish between separators, e.g. Problem P.2.

great circle L is a *bisector* of $\mathcal{P}$ if L is a separator of $\mathcal{P}$ and L divides $\mathcal{P}$ into two subsets $\mathcal{P}^+$ and $\mathcal{P}^-$ such that

$$| \text{card}(\mathcal{P}^+) - \text{card}(\mathcal{P}^-) | \equiv n \text{ (modulo 2)},$$

where card(·) denotes the cardinality of a set. The division of $\mathcal{P}$ induced by a bisector is called a bisection.

Problem P.2 (All Bisections). Given a set $\mathcal{P}$ of n polygons on the sphere, find all possible bisections of the set $\mathcal{P}$.

If a separator L divides $\mathcal{P}$ into two subsets $\mathcal{P}^+$ and $\mathcal{P}^-$, then

$$| \text{card}(\mathcal{P}^+) - \text{card}(\mathcal{P}^-) | = \left| \sum_{i=1}^{n} u_i(q) \right|,$$

where q is the dual point of L. From the definition of a bisector, an algorithm for finding all the bisectors is immediate by modifying Step 2 of Algorithm Find-All-Separators slightly.

Algorithm FIND-ALL-BISECTORS

Step 1. Apply Algorithm Compute-All-Sets-of-Equivalent-Great-Circles.

Step 2. Find all set of equivalent great circles that are bisectors. Traverse all sets of equivalent great circles starting from the set whose ownership vector was explicitly stored. Count the number of zero terms, k, and compute the

sum of all terms, $s = \sum_{i=1}^{n} u_i(q)$, in this ownership vector. If $k = 0$ and $|s| \equiv n$ (modulo 2), then the corresponding set of equivalent great circles are bisectors. During the traversal, k and s can be updated in constant time by using the face-adjacency graph.

The additional updating in Step 2 does not affect the overall time complexity which remains O(nv log n).

4.3. Minimally/Maximally-Cutting Great Circles

A generalization of Problem P.1 is the problem of finding the great circles that intersect the minimum number of polygons rather than those that does not intersect any polygon in $\mathcal{P}$. Clearly, if there exists a separator for the given set of polygons, then Problem P.3 reduces to Problem P.1.

Problem P.3 (Minimally Cutting Great Circles). Given a set $\mathcal{P}$ of n polygons on the sphere, find great circles that intersect the smallest number of the polygons in $\mathcal{P}$.

The opposite to Problem P.3 is the problem of finding the great circles that intersects the maximal number of the polygons in $\mathcal{P}$.

Problem P.4 (Maximally Cutting Great Circles). Given a set $\mathcal{P}$ of n polygons on the sphere, find great circles that intersect the largest number of the polygons.

This is a generalization of the problem of finding a stabbing line [2,8,9,11] which intersects a set of n polygons in the ensemble. In particular, Edelsbrunner, Guibas, and Sharir [9] gives an $O(v\,\alpha(v)\log v)$ time algorithm for constructing a representation of the stabbing lines of polygons with a total number of v vertices, where $\alpha(v)$ is the inverse of Ackerman's function. Hershberger's result in [11] reduces the time complexity of computing a representation of stabbing lines for a set of polygons to $O(v \log v)$ time. Obviously, if a stabbing great circle exists for the set of spherical polygons, then Problem P.4 reduces to the stabbing line problem. However, if a stabbing line does not exist, we may still want to determine which great circles intersect the largest number of polygons, as in the application of 4- and 5-axis numerical control machining. In such cases, naively applying the stabbing line algorithm to solve Problem P.4 would yield a time complexity of

$$T(n) + \binom{n}{n-1} T(n-1) + \binom{n}{n-2} T(n-2) + \cdots + \binom{n}{1} T(1),$$

in the worst case, where T(k) denotes the time required to compute the representation of a stabbing line of k of the n polygons.

We present an O(nv log n) algorithm for solving Problem P.4 as well as for Problem P.3. Let the set of given polygon be $\mathcal{P} = \{ P_i \mid i = 1,..., n \}$. Also, let a great circle be $L = \{ x \mid (q \cdot x) = 0, q,x \in S^2 \}$, i.e., q is the dual point of L. Now, from Corollary 3.1, we know that if the ownership vector u(q) has $u_i(q) = 0$, then L intersects polygons P_i. Thus, Problem P.3 and Problem P.4 can be solved by finding the sets of equivalent great circles with the minimum number and the maximum number, respectively, of the zero terms in their ownership vectors.

Algorithm MIN (MAX)-CUTTING-GREAT-CIRCLES

Step 1. Apply Algorithm Compute-All-Sets-of-Equivalent-Great-Circles.

Step 2. Traverse all sets of equivalent great circles starting from the set whose ownership vector is explicitly stored. Count the number of zero terms, k, in this ownership vector. While traversing the sets of equivalent great circles, update k in constant time by using the face-adjacency graph and keep track of the current minimum (or maximum) value of k and the set(s) of great circles realizing the optimum.

4.4. Densest Hemisphere

The previous problems are all concerned with the finding of optimal great circles. Of equal interest are problems that find optimal hemispheres, one of which is the following:

Problem P.5 (Densest Hemisphere). Given a set $\mathcal{P}$ of n polygons on the sphere, find a hemisphere that contains the largest number of the polygons in $\mathcal{P}$.

A similar problem concerning points on the sphere has been discussed by Johnson and Preparata [13], in which a set of v points is given and an $O(v^2 \log v)$ algorithm is reported for determining a densest hemisphere (i.e. a hemisphere that contains a largest subset of points)†. Here, we are given a set of n polygons on the sphere, and we present an O(nv log n) algorithm for determining a hemisphere that contains the largest subset of polygons, where v is the total number of vertices in the polygons.

An examination of the characteristics of the dual point of a densest hemisphere reveals the algorithm. Let $H^+ = \{ x \mid (q \cdot x) \geq 0 , q,x \in S^2 \}$ be a hemisphere, and q be its dual point. Now, from

† We note that the densest hemisphere of a point set can be determined in $O(v^2)$ time by using the algorithm described in [5].

Theorem 2.2, a spherically-convex polygon P is contained in H^+ if and only if q is in the interior of P*. This implies that if the ownership vector u(q) has terms $u_i(q) = 1$, $i \in \{1,..., n\}$, then H^+ contains the polygons P_i. Thus, H^+ is a densest hemisphere if u(q) has the largest number of 1's.

Algorithm DENSEST-HEMISPHERE

Step 1. Apply Algorithm Compute-All-Sets-of-Equivalent-Great-Circles.

Step 2. Traverse all sets of equivalent great circles starting from the set whose ownership vector was explicitly stored. Count the number of terms that equals to 1 in this ownership vector. Let this number be k. While traversing the sets of equivalent great circles, update k in constant time by using the face-adjacency graph and keep track of the current maximum value of k as well as a hemisphere realizing it.

5. SUMMARY

We have identified a set of computational geometry problems on the sphere: cutting spherical polygons with zero, minimal, and maximal intersections. These problems are shown to utilize the same struc-

ture – sets of equivalent great circles which partition the unit sphere. The solutions to these problems offer new benchmarks in computational complexity.

Perhaps equally satisfying, the solutions address unsolved problems in numerical control machining and coordinate measurement: the determination of workpiece orientation such that the number of setups is minimized on a 3-, 4-, or 5-axis machine with a ball-end tool can now be effected.

REFERENCES

[1] A.V. Aho, J.E. Hopcroft, and J.D.Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Mass., 1974.

[2] D. Avis and R. Wenger, *Polyhedral Line Transversals in Space*, Discrete and Computational Geometry, 1989.

[3] B. Baumgart, *A Polyhedron Representation for Computer Vision*, National Computer Conference, AFIPS Conf. Proc., 589-596, 1975.

[4] K.Q. Brown, *Geometric Transformations for Fast Geometric Algorithms*, Ph. D Thesis, Dept. of Computer Science, Carnegie Mellon Univ., Dec. 1979.

[5] B.M. Chazelle, L.J. Guibas, and D.T. Lee, *The Power of Geometric Duality*, BIT 25, 76-90, 1985.

[6] L.L. Chen and T.C. Woo, *Computational Geometry on the Sphere*, to appear in Transactions of ASME, Journal of Trans. , Mech. , and Automation in Design.

[7] K.L. Clarkson, *A Las Vegas Algorithm for Linear Programming When the Dimension is Small*, Proc. of IEEE Symposium on Foundations of Computer Science, 452-456, 1988.

[8] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer-Verlag, 1987.

[9] H. Edelsbrunner, L.J. Guibas, and M. Sharir, *The Upper Envelope of Piecewise Linear Functions: Algorithms and Applications*, Discrete and Computational Geometry, 4(4), 311-336, 1989.

[10] L.J. Guibas and J. Stolfi, Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagram, ACM Trans. on Computer Graphics, Vol.4, No. 2, Apr. 1985, 74-123.

[11] J. Hershberger, *Finding the Upper Envelope of n Line segments in O(n log n) Time*, Discrete and Computational Geometry, ***,1989.

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

[13] D.S. Johnson and F.P. Preparata, *The Densest Hemisphere Problem*, Theoretical Computer Science, 93-107, 1979.

[14] D.E. Knuth, *The Art of Computer Programming. Vol. I: Fundamental Algorithms*, Addison-Wesley, Reading, Mass., 1968.

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

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

[17] J. O'Rourke, C. Chien, T. Olson, D. Naddor, *A New Linear Algorithm for Intersecting Convex Polygons*, Computer Graphics and Image Processing 19, 384-391, 1982.

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

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

[20] R. Sedgewick, *Algorithms*, Addison-Wesley, 1983.

[21] T.C. Woo, *A Combinatorial Analysis of Boundary Data Structure Schemata*, IEEE Computer Graphics and Applications, 19-27, March, 1985.

APPENDIX. COMPUTING A SPHERICAL PARTITIONING BY CONVEX POLYGONS IN O(nv log n) TIME

Let $C = \{ C_1,..., C_n \}$ be a set of convex polygons on the sphere. These convex polygons partition the surface of the sphere into *faces*, each consisting of points covered by the same subsets of polygons in C. Since this partitioning is embedded on the sphere, we can represent it by using any of the data structures for planar graphs, such as the doubly-connected-edge-list data structure [18], the winged-edge data structure [3], the symmetric data structure [21], or the quad-edge data structure [10]. Here, we use the winged-edge data structure.

Each face can be a simple polygon or a polygon with holes, assuming that all vertices of polygons in C are in general position. (Figure A.1 illustrates a non-simple face.) Simple faces can be constructed from the intersection relationships among the convex polygons. To construct holes for non-simple faces, however, it is necessary to identify outer contours formed by sets of polygons in C, where each set consists of polygons that intersect at least one other polygon in the set. A set of such polygons is called a *connected component*. By definition, there is no intersection between polygons from two connected components. Thus, two connected components are either mutually exclusive or one contains the other. Consequently, containment relationships among the connected components can be represented by a

binary tree. Each connected component is represented by a node with two pointers: one points to a connected component mutually exclusive with it and the other points to a connected component contained by it. For example, let $\mathcal{K}_1$ and $\mathcal{K}_2$ be the two connected components which consist of three and two convex polygon respectively, as illustrated in Figure A.1. Then, in the containment tree, $\mathcal{K}_2$ points to $\mathcal{K}_1$ indicating containment. With these containment relationships, we can identify a non-simple polygonal face.

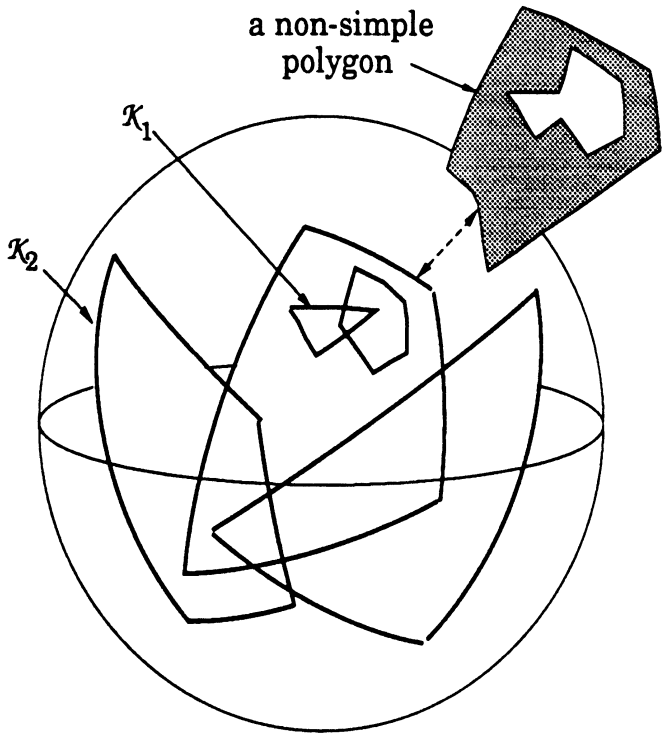

Figure A.1 A non-simple face in a partitioning

With this terminology, we present the algorithm for constructing the spherical partitions by a set of convex polygons:

Algorithm SPHERICAL-PARTITIONING-BY-CONVEX-POLYGONS

Input: A set of convex polygons $C = \{C_1,..., C_n\}$.

Output: A representation of the spherical partitioning by the polygons in C.

Step 1. Construct simple faces by utilizing intersection relationships among the polygons in C.

1.1 Compute the intersections among the polygons.

1.2 For each edge, sort the intersection points.

1.3 Merge the coincident intersection points.

1.4 Determine simple faces.

Step 2. Construct non-simples faces by utilizing containment relationships among the polygons in C.

2.1 Identify the connected components.

2.2 Establish containment relationships among the connected components.

We now describe the algorithm in detail and analyze the time complexity required of each step:

Step 1. (1.1) To identify all the intersection points, we find the intersection between each pair of convex polygons. Assuming that all the vertices are in general position, the intersection of two edges is a point which is recorded for both edges. Let v_i denote the number of

vertices of convex polygon C_i, i = 1,..., n and let $v = \sum_{i=1}^{n} v_i$. Then, we can identify all the intersection points in $\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} O(v_i + v_j) = O(nv)$ time by using the convex polygon intersection algorithm described in [17].

(1.2) For each edge, we sort the intersection points (by their parametric values). The intersection points split the original edges into a number of segments. Let these new edges inherit the orientations of the edges from which they are split. Since each convex polygon can intersect an edge at two points in the worst case, each edge can contain at most 2(n-1) intersection points. Therefore, we can finish the sorting in O(vn log n) time.

(1.3) We complete the intersection graph by merging coincident intersection points. At the same time, we sort the incident edges in counter-clockwise order for each intersection point. To analyze the time complexity required for the sorting, we observe that, for an edge with m distinct intersection points p_i, i = 1,..., m, we have $\sum_{i=1}^{m} e_i \leq 2(n-1)$, where e_i is the number of edges incident at p_i. (This is because each edge has at most 2(n-1) intersection points). Thus, we can complete sorting incident edges for all intersection points on an edge in O(n log n) time. (The worst case happens when (n–1) edges incident at a single point.) We therefore can merge coincident intersection points in O(vn log n) time.

(1.4) We determine simple faces by performing a traversal of the graph formed by the intersecting convex polygons. The traversal starting at an arbitrary vertex and stops when all vertices have been traversed. We can complete this traversal in O(nv) time as there are O(nv) edges in the graph.

Step 2. (2.1) We then identify the connected components. In (1.1), we have already identified the intersections between any two polygons in C. With this information, we can determine the connected components in $O(n^2)$ time by performing a depth-first-search on the intersection graph as described in [20].

(2.2) We establish the containment relationships among the connected components by utilizing the containment relationships among the polygons. Let $\mathcal{K}_1$ and $\mathcal{K}_2$ be two connected components. If a polygon $C_i \in \mathcal{K}_1$ contains another polygon $C_j \in \mathcal{K}_2$, then $\mathcal{K}_1$ contains $\mathcal{K}_2$. On the other hand, if neither $\mathcal{K}_1$ nor $\mathcal{K}_2$ contains each other, they are mutually exclusive. For any two non-intersecting convex polygons C_i and C_j, we can determine in $O(\log v_i + \log v_j)$ time whether C_i contains C_j, C_j contains C_i, or neither, by checking point inclusion [18]. Thus, we can determine such relationship between pairs of polygons from different connected components in $O(n^2 \log (\frac{v}{n}))$† time. We can then complete the containment tree time by applying topological sort [14]

---

† $\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} O(\log v_i + \log v_j) = (n-1) \sum_{i=1}^{n} O(\log v_i)$

$\leq (n-1)\, O(n \log (\frac{v}{n})) = O(n^2 \log (\frac{v}{n}))$

which takes $O(n^2)$ time. (In the worst case, the number of connected components is n.)

Thus, We have described an O(nv log n) time algorithm for computing the partitions of the sphere induced by a set of convex polygons.