ELSEVIER

Computer-Aided Design 35 (2003) 1269–1285

COMPUTER-AIDED DESIGN

www.elsevier.com/locate/cad

Maximal intersection of spherical polygons by an arc with applications to 4-axis machining

Kai Tang*, Yong-Jin Liu

Deptartment of Mechanical Engineering, Hong Kong University of Science and Technology, Hong Kong, People's Republic of China

Received 3 October 2002; received in revised form 6 March 2003; accepted 7 March 2003

Abstract

Many geometric optimization problems in CAD/CAM can be reduced to a maximal intersection problem on the sphere: given a set of N simple spherical polygons on the unit sphere and a real number constant $L \leq 2\pi$, find an arc of length L on the unit sphere that intersects as many spherical polygons as possible. Past results can only solve this maximization problem for two very restricted special cases: the arc must be either a great circle or a semi-great circle. In this paper, a simple and deterministic algorithm based on domain partitioning is presented for solving this maximal arc intersection problem in the general case when the number L is arbitrary. The algorithm is made possible by reducing the domain of the arcs to a continuous sub-space in $\mathbf{R}^2$ and then establishing a quotient space partitioning in this sub-space based on a congruence relation. The number of the constituting congruent sub-regions in this quotient space partitioning is shown to have an upper-bound $O(E^3)$, where E is the total number of edges on the polygons. The proposed algorithm has a worst-case upper bound $O(ME)$ on its running time, where M is an output-sensitive number and is bounded by $O(E^3)$. Examples including two realistic tests for 4-axis NC machining are presented.

Keywords: Spherical polygons; Maximal intersection; Free form surface machining; Domain partitioning; Computational geometry

1. Introduction

This paper studies the following geometrical optimization problem, referred to as the *maximal arc intersection* problem: given N simple spherical polygons on the unit sphere **S** that can overlap each other, and a real number $L \leq 2\pi$, find a great arc of length L on S that intersects a maximal number of the given spherical polygons. Throughout this paper, the term *great arc* refers to a segment of a great circle and is abbreviated as *arc*. In Fig. 1, an example of this optimization problem is given.

There is a number of industrial applications of this optimization problem; among them a particular one is the minimization of work-piece set-ups in 4-axis surface machining. Fig. 2(a) shows a schematic picture of a 4-axis numerically controlled (NC) machine. The tool moves in three principal directions X, Y, Z and the work table rotates about a fourth axis which is usually parallel to one of the three principal axes (x-axis in this case). A set-up refers to a placement of the part on the worktable with a fixed orientation with respect to the tool and the worktable. The minimization of set-ups then refers to the careful selection of set-ups so that the entire part surface can be correctly machined without any gouging and with as few set-ups as possible. To put this minimization problem in a more computational viable format, the part surface is usually first partitioned into a number of faces $f_1, f_2, \ldots, f_N$, with each face f_i associated with a set of directions, called *accessible orientations*, along which the tool can access any point on the face without interfering with other faces. The set of accessible orientations of face f_i is best represented as a spherical polygon V_i on the Gaussian sphere (i.e. the unit sphere **S**,), called the *visibility map* [8,9]. For a particular set-up of the machine, the orientations of the tool as provided by the rotation of the work table form an arc of length θ_r on **S**, where θ_r is the allowable range of rotation angle of the fourth axis which is at most π (notice that the work table is usually flat). Obviously, as illustrated by Fig. 2(b), the face f_i is accessible (machinable) in a set-up if and only if the representative arc of that set-up and

* Corresponding author.
E-mail addresses: mektang@ust.hk (K.Tang); liuyj@ust.hk (Y. -J. Liu).

0010-4485/03/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/S0010-4485(03)00042-3

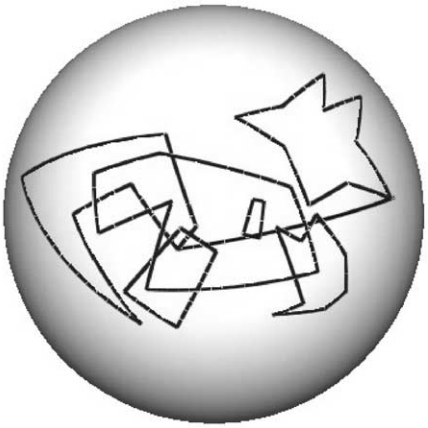


Fig. 1. Maximal intersection of spherical polygons by an arc of fixed length; the optimal arc is shown in blue color.

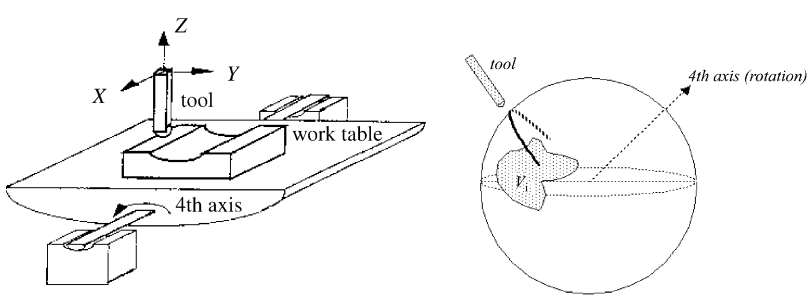
the visibility map V_i have at least one common point. To minimize the number of set-ups is then equivalent to finding a minimum set of arcs (of length θ_r) such that any f_i is intersected by at least one of the arcs. Since this minimization is easily seen to be NP-hard [10], heuristic alternatives have been sought to search for near-optimal solutions. Among them a promising one is the *iterative greedy* approach: at each iteration, an arc (of length θ_r) is sought that intersects a maximum number of the given V_i; these intersected V_i are then removed from the set of visibility maps for the next iteration; the iteration continues until the set of visibility maps becomes empty. Obviously, at each iteration, it is exactly the maximal arc intersection problem.

2. Prior work and contribution of this paper

Due to the difficulty of the search domain being a continuous sub-region in **S**, earlier work can only solve the maximum arc intersection problem in a very restricted form: the length of the arc must be either exactly 2π, i.e. a great circle, or exactly π, i.e. a semi-circle. If the arc length is limited to 2π, Tang et al. [13] used the central projection technique to transform the problem from the sphere to the plane and devised an $O(NE \log E)$ algorithm to solve the problem based on the partitioning scheme, where E is the total number of edges on the given N spherical polygons. Utilizing the idea of duality transformation [1,2], later Gupta et al. [11] presented an algorithm that runs in $O(E^2)$ time. If the arc length is limited to π, based on both central projection [6] and duality transformation, Tang et al. [14] were able to give an $O((E + I_{wb})^2 N)$ time algorithm that finds a semi-circle on **S** intersecting the maximal number of the given spherical polygons, where I_{wb} is the number of intersections between the edges of the image polygons under central projection of the spherical polygons belonging to the upper- and lower-sphere, respectively.

From the practical point of view, as already alluded earlier, the allowable range of rotation θ_r is always strictly less than π; therefore the prior results do not provide the desired optimal solution. In terms of theoretical significance, an exact algorithmic solution handling the most general case of an arbitrary arc other than a semi-circle or great circle will further the research and solve this spherical optimization problem. This paper achieves this objective. Specifically, a simple and deterministic algorithm is presented that finds an exact solution to the maximal arc intersection problem for an arbitrary arc length L. This solution is important both from application point of view [8] and from computational geometry point of view [4].

The paper is organized as follows. In Sections 3 and 4, for a clear description of the algorithm, we first offer detailed geometric analyses and the algorithmic solution to the analogue of the maximal arc intersection problem in the plane: find a line segment of a specified length L that intersects a maximal number of the given polygons in the plane. We then in Section 5 show how the algorithm developed for the planar case can be transformed to the spherical domain. A set of experiments, including two examples in NC machining, are presented in Section 6 to demonstrate the algorithm. Finally, we conclude the paper in Section 7 with a discussion on some future research topics.



(a) (b)

Fig. 2. (a) A 4-axis NC machine; (b) visibility maps.

3. Analytic structure in the plane

We first consider the counter-part of the maximum arc intersection problem in the plane: given a set of simple polygons in the plane and a constant real number L, how to find a line segment s of length L that intersects a maximum number of the polygons? Fig. 3 shows such an example. That we analyze the problem first in the plane but not on the sphere is due to an obvious consideration: it is usually easier and more discernable to describe geometrical entities and relationships in the plane than in the spherical domain. By first conducting the analysis and solving the problem in the plane and then providing a rigorous one-to-one mapping of the structure of the solution from the plane to the sphere, our original problem on the sphere is readily solved. In this section, we conduct the analysis on the geometrical and combinatorial structure of the problem in the plane, while its algorithmic details are left to Section 4.

Let $s(p, \theta)$ represent an arbitrary line segment of length L in the plane, where p is one end point of the segment and θ is the angle measured counter-clockwise from the $+x$-axis to the vector pointing from p to the other end point of the segment. The number of the polygons intersected by $s(p, \theta)$ will be called its *size*. We begin with a simple observation.

Lemma 1. *If an $s(p,\theta)$ has size k, then there exists an $s(p',\theta)$, with one of its two end points lying on an edge of a polygon, whose size is at least k.*

Proof. Without loss of generality, suppose $s(p, \theta)$ intersects polygons $P_1, P_2, \ldots, P_k$. One can then translate $s(p, \theta)$ in the θ direction until one of its two ends touches an edge of some polygon P_i. The new line segment $s(p', \theta)$ either intersects the same set of polygons as s (p, θ) if P_i is one of $\{P_1, P_2, \ldots, P_k\}$ or $k + 1$ polygons otherwise. □

With Lemma 1, the search domain thus has been reduced to those $s(p, \theta)$ whose end points lie on some edges of polygons. We will say $s(p, \theta)$ *sits* on an edge e if $p \in e$. Consider an arbitrary edge e of a polygon whose length is h. Without loss of generality, suppose this edge has an inclination angle of 0 degree, that is, it lies on the x-axis, and its left end point is the origin of the coordinate system. Any point p on e can be expressed uniquely as $p = (t, 0)$, $(0 \le t \le h)$. Consequently, an $s(p, \theta) : p \in e$ is a function $s(t, \theta)$ of two parameters t and θ with a range of $[0, h]$ and $[0, 2\pi]$ respectively. For the purpose of clearer discussion, and also to better suit the need of easy conversion from the plane to the sphere, the $[0, 2\pi]$ range for θ is divided into two halves $[0, \pi]$ and $[\pi, 2\pi]$ and they will be dealt with separately. The analysis now becomes: how to partition the $t - \theta$ domain $[0, h] \times [0, \pi]$ into a finite number of sub-regions so that each of these sub-regions bears a similar solution structure.

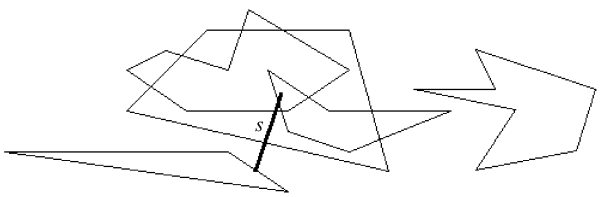


Fig. 3. Maximal intersection of polygons by a line segment s of fixed length.

3.1. Characteristic and eigen lists

Consider a point $(t, 0) : t \in [0, h]$ and its relationship with an arbitrary line segment u with respect to an $s(t, \theta)$. Since the range for θ is limited to $[0, \pi]$, the segment u is assumed to lie entirely in the half-space $y \ge 0$. (If an edge of a polygon crosses the x-axis, only its portion above the x-axis will be considered.) We say that u *covers* point $(t, 0)$ (or point t for brevity) if there is at least one point $q \in u$ such that the distance between the two points q and $(t, 0)$ is less than or equal to L. The *covering set* of t then refers to the set of the edges of the polygons that cover t. To a u that covers t, let σ_u be the subset in $\mathbf{R}$ such that if an $s(t, \theta)$ intersects u then $\theta \in \sigma_u$, and vice versa. Due to the linearity of u, it can be easily seen that σ_u must be in the form of a closed interval $[\theta_i, \theta_o]$ in $\mathbf{R}$. $[\theta_i, \theta_o]$ will be referred to as the *covering interval* of u at parameter t, and θ_i and θ_o the *bounding angles*. In addition, θ_i will be called the *in-angle* and θ_o the *out-angle* of the interval.

Now, let $u_1, u_2, \ldots, u_m$ be the covering set of t. One can sort the in-angles and out-angles of their covering intervals into an ordered list of $2m$ numbers $\{\theta_1, \theta_2, \ldots, \theta_{2m}\}$. This list will be defined to be the *characteristic list* of t, denoted as $Cl(t)$. By definition of a covering interval, for any interval $[\theta_i, \theta_{i+1}]$ $(i = 1, \ldots, 2m - 1)$, all the $s(t, \theta) : \theta \in (\theta_i, \theta_{i+1})$ intersect the same set of line segments. Consequently, the list $\{\theta_1, \theta_2, \ldots, \theta_{2m}\}$ introduces another list of $2m - 1$ sets $\{\xi_1, \xi_2, \ldots, \xi_{2m-1}\}$, where each ξ_i is the set of line segments that any $s(t, \theta) : \theta \in (\theta_i, \theta_{i+1})$[1] intersects. This second list at t will be defined as the *eigen list* of t, with the notation $Ei(t)$. In Fig. 4, an example is given to illustrate these two lists. As we will see next, although list $Cl(t)$ completely determines list $Ei(t)$, it is the latter that solely decides the structure of the solution space which is the key in discretizing the search domain.

3.2. Analyses of critical points

Given two different points t and t', their characteristic lists $Cl(t)$ and $Cl(t')$ are different. On the other hand, their eigen lists $Ei(t)$ and $Ei(t')$ may or may not be identical to

[1] We purposely avert the boundary case of $s(t, \theta_i)$. It can be easily shown however that the set of line segments that $s(t, \theta_i)$ intersects is either ξ_i or ξ_{i+1}.

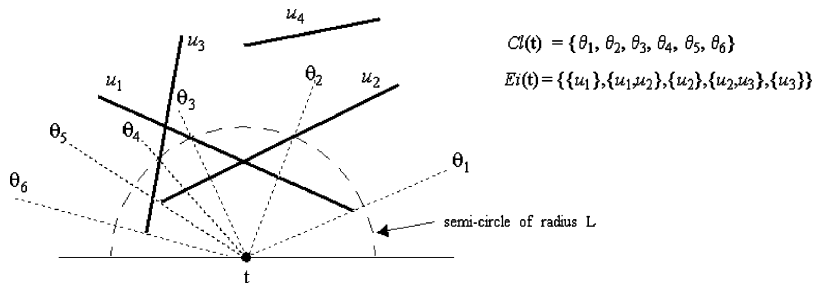


Fig. 4. Characteristic and eigen lists.

each other. Since two identical $Ei(t)$ and $Ei(t')$ give identical structure of solutions, we investigate under what conditions can two lists $Ei(t)$ and $Ei(t')$ be identical to each other. Let's define below a congruence relation between two points t and t' based on their eigen lists.

Congruence. Two points t and t' are said to be *congruent* to each other if and only if their eigen lists $Ei(t)$ and $Ei(t')$ are identical to each other.

Next, let's consider the 'difference' between two characteristic list $Cl(t)$ and $Cl(t')$. Unlike $Ei(t)$, however, since a $Cl(t)$ contains structure information more than just identifiers, such as in-angles and out-angles, a quantitative and structural measure is needed to describe the similarity between $Cl(t)$ and $Cl(t')$. The following definition is in order.

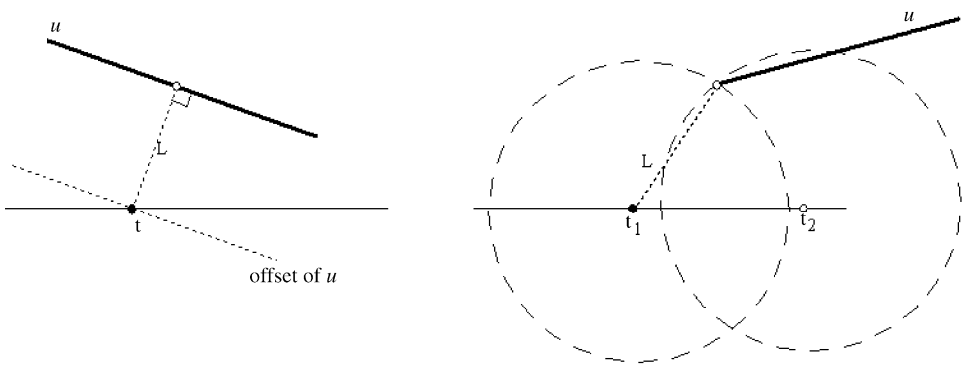
Equivalence. Two characteristic lists $Cl(t) = \{\theta_1, \theta_2, \ldots, \theta_{2m}\}$ and $Cl(t') = \{\phi_1, \phi_2, \ldots, \phi_{2n}\}$ are said to be *equivalent* to each other if and only if the following two conditions are both satisfied:

(1) t and t' have the same covering set (which means $m = n$), and (2) θ_i is the in-angle (out-angle) of the covering interval at t of a line segment u if and only if ϕ_i is the in-angle (out-angle) of the covering interval at t' of the same line segment u, for $i = 1, 2, \ldots, m$.

It is conceivable that for a small perturbation δ in t, the two characteristic list $Cl(t)$ and $Cl(t + \delta)$ should be equivalent to each other; however, there are certain *critical* points at which the nature of the characteristic list changes suddenly. We consider under what circumstances such changes occur. They are categorized into six cases under two types, based on the way the two conditions in the definition of equivalence relation are affected.

3.2.1. Type I. Emerging or disappearing of a covering line segment

This type of change refers to the situation when the covering interval of a line segment u enters or leaves a $Cl(t)$; or in other words, when equivalence condition (1) is becoming unsatisfied. Geometrically, due to the linearity of u, this means that there must exist a positive real number σ such that u either covers $[t - \sigma, t]$ and does not cover any point in $(t, h]$ or covers $[t, t + \sigma]$ but none of $[0, t)$. This geometrical observation implies the *circular exclusion* property: t is a type I critical point due to u if and only if the circle of radius L and centering at t intersects a single point of u and contains no point of u in its interior. Referring to Fig. 5, there can only be two cases for such a circle: it either touches an interior point of u (Fig. 5(a)) or passes through an end point of u and excludes the rest of u. Both cases can be identified geometrically. In the first case, t is



(a) Type Ia (b) Type Ib

Fig. 5. Type I critical points.

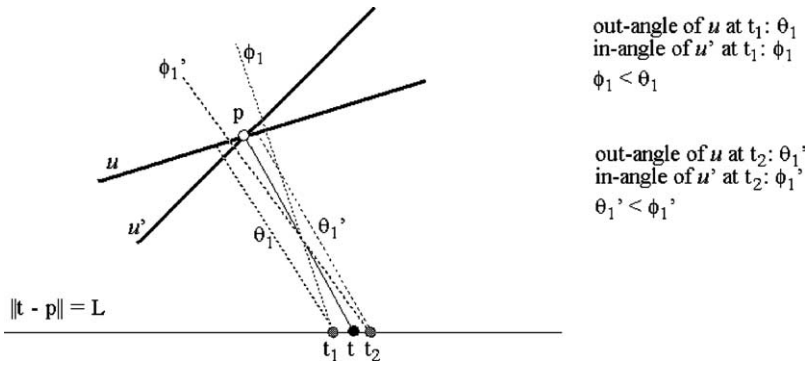


Fig. 6. Type IIa critical point.

the sole intersection point, if it exists, between the interval $[0, h]$ (on the x-axis) and the offset of u, as shown in Fig. 5(a), where the offset distance is L. The latter case can also be easily identified by intersecting the interval $[0, h]$ with the circle of radius L and centering at an end point of u, as in Fig. 5(b) (notice that only t_1 is a type I point, but not t_2).

3.2.2. Type II. Order change of covering intervals

This second type pertains to the circumstance when equivalence condition (2) is about to be jeopardized. Because a bounding angle in a $Cl(t)$, which is the slope angle of a vector from t to either a line segment's endpoint or a circle-line intersection point, is a continuous and smooth function of t between critical points, a critical point t pertaining to condition (2) should be the one at which two bounding angles, each belonging to a different covering interval, become coincidental. In other words, t is a type II critical point if in its characteristic list $Cl(t) = \{\theta_1, \theta_2, \ldots, \theta_{2m}\}$ there are two identical bounding angles $\theta_i = \theta_{i+1}$ for some i and they belong to different covering intervals. The only exception to the above assertion is the degenerate case when one end point of a line segment u lies on the edge e; in this case the two bounding angles of u changes drastically when crossing the point. Depending on the contributing sources to the critical point, four sub-types are further defined.

3.2.2.1. Type IIa. Common interior point case. This first sub-type refers to the case when a type II critical point is contributed by a single point. Suppose two line segments u and u' intersect each other at a point p. Let t be a point in the interval $[0, h]$ whose distance to p is the length L, if it exists. Ignoring the degenerate case when the line segment $[p, (t, 0)]$ is perpendicular to one of u or u', there must exist $t_1 < t$ and $t_2 > t$ such that both u and u' cover $[t_1, t]$ and $[t, t_2]$. Referring to Fig. 6, the two adjacent bounding angles in $Cl(t_1)$, each belonging to u and u', respectively, will switch their order in the characteristic list $Cl(t_2)$. In Appendix A, a formal proof is given on this switching of order. Computationally, t is obtained by intersecting the circle centering at p and of radius L with the line segment $[(0, 0), (h, 0)]$; at most two such critical points exist for a common point p.

3.2.2.2. Type IIb. Interior point + end point case. In this scenario, the critical point is contributed by one interior point of u and one end point of u'. As illustrated in Fig. 7, in a neighborhood of t clear of other critical points, two bounding angles, one belonging to u and the other belonging to u', will switch their order in the characteristic list when at different sides of t. Its proof is similar to that of type IIa and is omitted here.

Geometrically, t is determined by locating an interior point p on u such that the distance between p and point $(t, 0)$

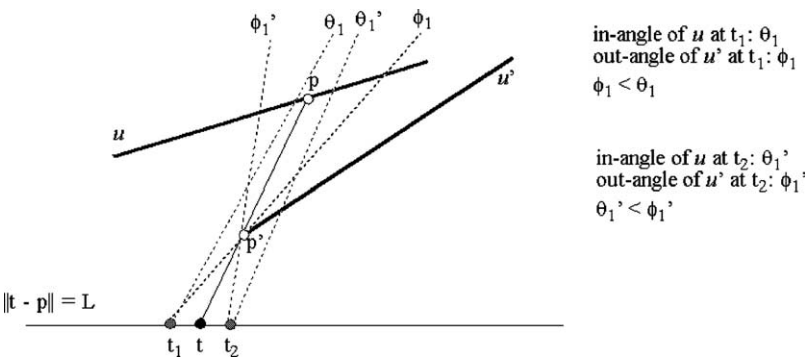


Fig. 7. Type IIb critical point.

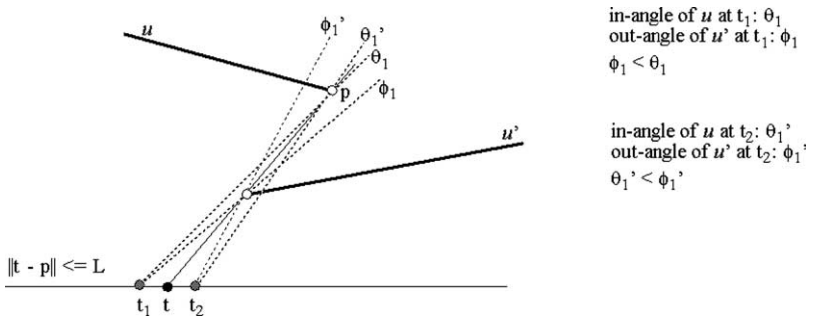


Fig. 8. Type IIc critical point.

is L and one end point of u' lies in the interior of the line segment between p and $(t,0)$. Let (x_0,y_0) and (x_1,y_1) be the two end points of u and (x',y') be the end point of u' in interest. The following two degree-2 equations establish this geometrical relationship:

$$(x_0(1-v)+x_1v-t)^2+(y_0(1-v)+y_1v)^2=L^2$$

$$(x_0(1-v)+x_1v-t)/(y_0(1-v)+y_1v)$$

$$=(x_0(1-v)+x_1v-x')/(y_0(1-v)+y_1v-y').$$

The solution to the above two equations for the two variables v and t, with their ranges limited to [0, 1] and $[0,h]$ respectively, will give out at most two type IIb critical points.

3.2.2.3. Type IIc. End point + end point case. This sub-type of continuous bounding angles case covers the special configuration when a line through the end points of two different line segments u and u' intersects the segment $[0,h]$, as shown in Fig. 8. Notice that for a t to qualify to be a type IIc point, besides being collinear with the end points of u and u', it needs further to satisfy the distance constraint: the distances from $(t,0)$ to the two end points should both be no greater than L. We omit the proof of the order switching of the bounding angles in this case, as it is similar to that of type IIa which is given in Appendix A.

3.2.2.4. Type IId. End point degenerate case. This last type corresponds to the special situation when an end point of a line segment u falls on the edge e. (Note that if the original u strictly intersects e, it is cut by the line $y=0$ into two halves.) Fig. 9 shows an example exemplifying the effect of this critical point on the characteristic list. As revealed in the example, the bounding angles of u changes drastically from $\{\theta_1,\theta_4\}$ to $\{\phi_3,\phi_4\}$ after crossing the critical point t.

3.3. Congruent partitioning

Let $\{t_1,t_2,\ldots,t_N\}$ be the critical points on the edge $[(0,0),(h,0)]$ as contributed by the edges of the N given polygons, sorted from left to right. Since the equivalence relation among characteristic lists implies the congruence relation of *the* corresponding points on the edge $[(0,0),(h,0)]$,the list$\{t_1,t_2,\ldots,t_n\}$ partitions the edge into $N+1$ congruent intervals (t_i,t_{i+1}), $i=0,1,2,\ldots,n$, with $t_0=0$ and $t_{n+1}=h$. As an illustration, Fig. 10 depicts such a partitioning for a set of four line segments. By definition, any two points p and q in a congruent interval will have an identical eigen lists, i.e. $Ei(p)=Ei(q)$. Two points p and q are said to have the same *solution structure* in the sense that if there is a line segment $s(p,\theta)$ for some $\theta\in[0,\pi]$ that intersects a subset K of the polygons then there must exist some $\theta'\in[0,\pi]$ such that the segment $s(q,\theta')$ intersects the same K, and vice versa. Though not very obvious, it can be shown that all the points in an open congruent interval will have a same solution structure. Consequently, a congruent interval can be represented by an arbitrary point in it.

A deeper insight of this congruent partitioning leads to the concept of the quotient space partitioning [7] in the $t-\theta$ space. As already discussed earlier, in an interval (t_j,t_{j+1}),

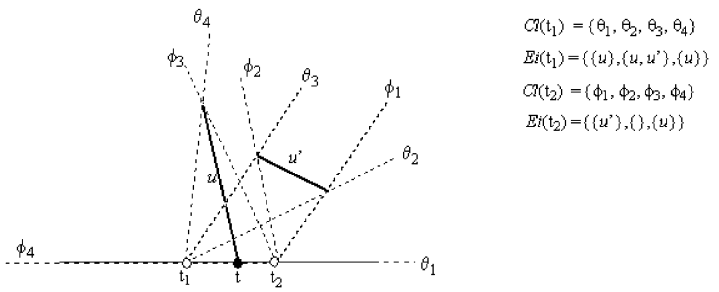


Fig. 9. Type IId critical point t.

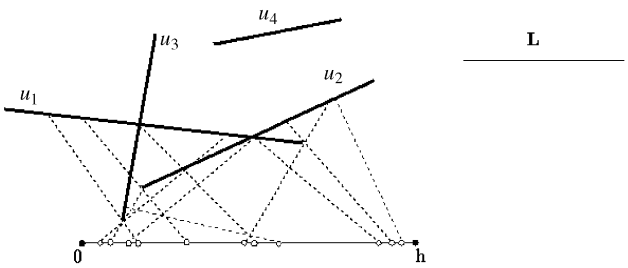


Fig. 10. Congruent partitioning on an edge.

the characteristic list $Cl(t) = \{\theta_1, \theta_2, \ldots, \theta_{2m}\}$ is continuous in t; that is, every θ_i is a continuous function of t, i.e. a smooth curve $\theta_i(t)$. These smooth curves $\theta_i(t)$ together with lines $t = t_j$ $(j = 1, 2, \ldots, n)$ then form a partitioning of the $t-\theta$ space in the domain $[0, h] \times [0, \pi]$, as schematically illustrated in Fig. 11(a). This is a quotient space partitioning in the sense that for any two points (t, θ) and (t', θ') belonging to a same sub-region in the partitioning, the sizes of $s(t, \theta)$ and $s(t', \theta')$ are equal to each other. As a result, each sub-region can be represented by an arbitrary point in it and a dual graph (also called the Reeb graph, cf. [7]) can be established: every node in the graph represents a unique sub-region in the partitioning and two nodes are connected by an edge if the corresponding sub-regions are neighbors. For instance, Fig. 11(b) depicts the Reeb graph of the quotient space partitioning in Fig. 11(a). We do not explicitly build this Reeb graph though. The task instead now becomes how to design an efficient algorithm to implicitly traverse this Reeb graph so that a node with the maximal size can be identified. This is the task of the next section.

4. Algorithmic details in the plane

With the congruent partitioning relation defined and established on an edge, the next task is to design simple and efficient algorithms to find a maximal intersecting line segment based on the congruent relation. The first and also the crucial step is to have a simple and proper data structure for a characteristic list $Cl(p)$. Let $\{e_1, e_2, \ldots, e_E\}$ be the edges on the N given polygons $P_1, P_2, \ldots, P_N$. For simplicity of discussion, let us assume that $Cl(p)$ is in the general state, that is, no bounding angles in $Cl(p)$ are the same, and p lies on only one edge, say e_1. A $Cl(p)$ is represented by an array $Cl\_array[1 : m]$ of records of five fields each. An element $Cl\_array[i](1 \le i \le m)$ carries the following information: Cl_array[i].angle: the value of the corresponding bounding angle; Cl_array[i].type: a tag that indicates if the bounding angle is 'in' or 'out'; Cl_array[i].edge: an integer that identifies the covering edge, e.g. 4 means the covering edge is e_4; Cl_array[i].polygon: an integer identifying the polygon that the covering edge belongs to; and finally Cl_array[i].number: an integer which is the number of *polygons* that any line segment $s(t, \theta) : \theta \in (Cl\_array$[i].angle, Cl_array[i + 1].angle) intersects. The algorithm given below outlines how a characteristic list $Cl(p)$ is constructed. Without loss of generality, assume that edge e_1 is collinear with the x-axis. In the routine, we use $Cir(p, L)$ to denote the upper semi-circle with the center at p and of radius L; a utility function Poly_Id(e_i) is also used which returns the ID of the polygon to which the edge e_i belongs.

Algorithm **Characteristic_list** (p)
Begin
Step 1.

$m = 0$;

Step 2.

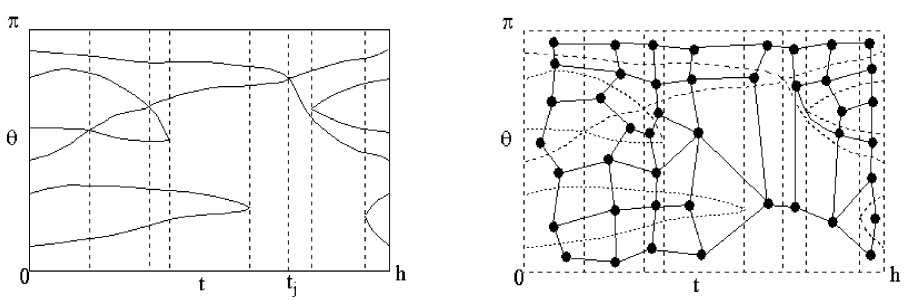
For $i = 2$ to E do begin
$\xi \leftarrow$ the portion of the line of e_i inside the semi-circle $Cir(p, L)$;
If $\xi \neq$Null then begin

$\sigma \leftarrow e_i \cap \xi$;

end
If $\sigma \neq$Null then begin

$m = m + 1$;

$Cl\_array\ [M]$.angle $\leftarrow$ the inclination angle of the vector from point p to the first end point of σ;
$Cl\_array\ [M + 1]$.angle $\leftarrow$ the inclination angle of the vector from point p to the second end point of σ;
If $Cl\_array\,[M]$.angle $>$ $Cl\_array\,[M + 1]$.angle then



(a) (b)

Fig. 11. A quotient space partitioning and its corresponding Reeb graph.

```
                Switch Cl_array[M].angle with Cl_array
                [M + 1].angle;
            Cl_array[M].type ← "in";
            Cl_array[M + 1].type ← "out";
            Cl_array[M].edge ← Cl_array[M + 1].edge ← e_i;
            Cl_array[M].polygon ←
            Cl_array[M + 1].polygon ← Poly_Id(e_i);
            Cl_array[M].number ←
            Cl_array[M + 1].number ← 0;
            m ← m + 1;
        End;
    End;

    Step 3.

    Sort the array Cl_array [1 : M] in ascending order
    according to the "angle" field;

    Step 4.1.

    For i = 1 to N do begin
        If (P_i strictly contains the point p) then begin
            Poly_register[i] ← 1;
            current_number ← current_number + 1;
        End
        Else
            Poly_register[i] ← 0;
    End;

    Step 4.2.

    Poly_register [Poly_Id(e_1)] ← 1;
    current_number ← current_number + 1;

    Step 5.

    For i = 1 to M − 1 do begin
        If Cl_array[i].type = "in" then begin
            If Poly_register[Cl_array[i].polygon] = 0 then
                current_number ← current_number + 1;
            Cl_array[i].number ← current_number;
            Poly_register[Cl_array[i].polygon] ←
            Poly_register[Cl_array[i].polygon] + 1;
        End
        Else begin
        Poly_register[Cl_array[i].polygon] ←
        Poly_register[Cl_array[i].polygon] − 1;
        If Poly_register [Cl_array[i].polygon] = 0 then
            current_number ← current_number − 1;
        Cl_array[i].number ← current_number;
        End;
    End;
End.
```

In the above algorithm, at Step 2, each edge e_i is checked to see if it intersects the upper semi-disc of radius L centering at p. If yes, the portion of e_i inside this semi-disc (which is a line segment denoted by σ in Step 2) will contribute two entries to the array *Cl_array*, each by an end point of σ; they are appended to *Cl_array* with their 5-fields records {angle, type, edge, polygon, number} set or initialized accordingly. The entries in the array *Cl_array* are then sorted based on their angle fields, at Step 3. Since a polygon can have more than one edge covering point p, and also counting those polygons that strictly contain p, a register *Poly_register* is maintained for each and every polygon. Basically, *Poly_register*[i] stores the number of edges of polygon P_i that are 'intersected' by the current interval (*Cl_array*[i].angle, *Cl_array*[i + 1].angle) when processed in Step 5; with an additional "1" added if P_i contains point p (either on the boundary or in the interior). At Step 4.1, *Poly_register* for every polygon P_i is initialized: it is set to "1" if P_i strictly contains the point p, or "0" otherwise. At Step 4.2 the *Poly_register* that corresponds to the polygon of edge e_1 is then set to 1, since p sits on e_1. The variable *current_number* is the number of polygons that the current interval (*Cl_array*[i].angle, *Cl_array*[i + 1].angle) (when processed in Step 5) 'intersects'. Initially, before the first angle *Cl_array*[1], *current_number* should be set to account only for those polygons that contain p (on the boundary or in the interior); this is done at Step 4. Actually, this initial value of *current_number* has an important meaning: it is the number of polygons that any $s(p, \theta) : \theta \in [0, Cl\_array[1]) \cup (Cl\_array[m].\text{angle}, \pi]$ intersects.

When processing a bounding angle *Cl_array*[i] in Step 5, we first increment or decrement by one the register *Poly_register* of the edge identified by *Cl_array*[i].edge, depending on whether it is an "in" or "out" edge. The number *Cl_array*[i].number is then updated from the previous interval *Cl_array* [i].number (which is also the value of the variable *current_number*) only if *Poly_register* changes from 0 to 1 or vice versa. This combined usage of the two variables *current_number* and *Poly_register* ensures that the number of intersected polygons is properly counted.

Let $s_{\max}(p)$ denote a line segment $s(p, \theta_0)$ with θ_0 belonging to one of the constituent intervals in $Cl(p)$ that has the largest "number" value. The lemma 2 below is in order.

Lemma 2. *It takes* $O(E + m \log m)$ *time and* $O(E)$ *space to construct the characteristic list* $Cl(p)$ *and find the line segment* $s_{\max}(p)$, *with* m *being the number of covering intervals at* p *which is at most* $2(E - 1)$.

Proof. By examining the 'number' field of the *Cl_array* in $Cl(p)$, the $s_{\max}(p)$ can be readily identified. Step 2 is easily seen to take $O(E)$ time. The sorting operation at Step 3 is $O(m \log m)$. The dominant operation of Step 4 is at Step 4.1 which can be achieved by a linear scanning of all the edges in regarding to the point p, hence

requiring O(E) time. The loop at Step 5 is trivially seen to take linear O(m) time. The proof for the space requirement is omitted. □

Next, let $\{c_1, c_2, \ldots, c_n\}$ be the congruent intervals induced by the critical points on an edge e. By definition of the congruence, in any c_i, for all the points $p \in c_i$, $s_{\max}(p)$ intersects the same number of polygons. Suffice it to say, if we pick an arbitrary point p_i *for* c_i, $i = 1, 2, \ldots, n$, and select among them the $s_{max}(p_k)$ that intersects the largest number of polygons, then the line segment $s = s_{\max}(p_k)$ should be a solution to our maximum intersection problem when the search domain $t \times \theta$ is restricted to $[0, h] \times [0, \pi]$ on edge e (which is assumed to be on the x-axis and have length h). To cover the other half of θ, i.e. $[\pi, 2\pi]$ we can reverse the direction of the y-axis and apply the same process on edge e to find another line segment s' that is the solution in the search domain $t \times \theta = [0, h] \times [\pi, 2\pi]$. Obviously, the one of s and s' that intersects a larger number of polygons must be a solution to the maximum intersection problem for the search domain $t \times \theta = [0, h] \times [0, 2\pi]$. The following lemma summarizes this result.

Lemma 3. *To a given arbitrary edge e, it takes O(E^3 log E) time and O(E^2) space to find a line segment s of length L among all the line segments of length L that sit on e such that s intersects a maximal number of the polygons.*

Proof. It is not hard to show that the maximal number of type I critical points that an edge can contribute to e is 2, and the number of any of the four type II critical points that a pair of edges can contribute is at most 4. Therefore, there are O(E^2) critical points on e which need to be sorted to obtain the congruent partitioning. Since it takes a constant time to calculate a critical point (referring to Figs. 5–9) and consider that all the intersection points between the edges of the polygons can be obtained in O(E^2) time, the O(E^2) congruent intervals can thus be computed in O(E^2 log E) time. For each congruent interval, we need to construct a $Cl(p)$ to find the line segment $s_{\max}(p)$ which takes O(E log E) time according to Lemma 2. Therefore O(E^3 log E) time is needed to go through all the congruent intervals and find the best solution $s_{\max}(p)$. Again, the analysis for the space requirement is omitted. □

It is natural to inquire if the upper-bound O(E^3 log E) in Lemma 3 can be improved. Observing that the difference between the sizes of $s_{\max}(p) : p \in c_i$ and $s_{\max}(p') : p' \in c_{i+1}$ for any two adjacent congruent intervals c_i and c_{i+1} in an edge e is at most one, it is inviting to ask whether the characteristic record for an congruent interval c_{i+1} can be obtained by incrementally *updating* the characteristic list of the previous congruent interval c_i. Let t_0 be a type I critical point due to an edge u that separates c_{i+1} from c_i, t_1 and t_2 be two points in c_i and c_{i+1} respectively. Suppose that $Cl(t_1) = \{\theta_1(t), \theta_2(t), \ldots, \theta_m(t)\}|_{t=t_1}$ each $\theta_i(t)$ being a continuous function of $t \in [t_1, t_2]$. (Note that the crossing of t_0 by t does not effect $\theta_i(t)$, since they are *not* contributed by u.) To get $Cl(t_2)$ from $Cl(t_1)$, we need to insert the covering interval of u at t_2 into the list $\{\theta_1(t_2), \theta_2(t_2), \ldots, \theta_m(t_2)\}$. Evaluating $\theta_i(t_2)$, $i = 1, 2, , \ldots, m$, takes a total of O(m) time, which is at most O(E); inserting an interval into the sorted list $\{\theta_1(t_2), \theta_2(t_2), \ldots, \theta_m(t_2)\}$ requires only O(log m) time, by using a balanced binary search tree. Once inserted, the other data such as the 'number' field of the new interval in $Cl(t_2)$ can be readily derived from the data of its preceding interval in the list. Therefore, it only takes O(m) time to obtain $Cl(t_2)$ from $Cl(t_1)$. Considering that m is usually much smaller than E, this is a not small saving from the tight O($E + m$ log m) bound if $Cl(t_2)$ is constructed from scratch (notice that algorithm **Characteristic_list**(p) has a lower bound of $\Omega(E)$ due to Step 2). Similar analyses can also be conducted on type IIa, type IIb, and type IIc critical points. The lemma given below summarizes these analyses (Lemma 4).

Lemma 4. *For a given arbitrary edge e, it takes* O($E^2 + n$ log $n + \sum_{i=2}^{n} m_i + k_e E$ log E) *time to find a line segment s of length L among all the line segments of length L that sit on e such that s intersects a maximal number of the polygons, where n is the number of congruent intervals on e which is bounded by* O(E^2), m_i *is the number of edges that cover the* ith *congruent interval which is at most* $E - 3$, *and* k_e *is the number of type IId critical points on e which is bounded by* O(E).

Proof. Referring to the proof of Lemma 3, it takes O($E^2 + n$ log n) time to establish the n congruent intervals on e. After constructing the characteristic record for the first congruent interval which takes O($E + m_1$ log m_1) $\leq O(E$ log E) time, the characteristic record for the ith congruent interval can be obtained from that of the preceding congruent interval in O(m_i) time. The last item in the expression, $k_e E$ log E, is due to the type IId critical points where a new characteristic list has to be built from scratch due to the drastic change of some bounding angles. □

Finally, by applying Lemma 4 to every edge, we can identify an edge e among all the E edges and a point p on e such that $s_{\max}(p)$ intersects a maximal number of polygons. Because of Lemma 1, it is immediately concluded that $s_{\max}(p)$ is also an solution to our maximum intersection problem. The theorem below summarizes this final result.

Theorem 1. *Given a set of polygons in the plane with a total of E edges and a real number L, we can in* O($E^3 + M \times$ log $M + EM + IE$ log E) *time and* O(E^2) *space find a line segment of length L that intersects a maximal number of the polygons, where M is the number of critical points on all the edges of the polygons and is bounded by* O(E^3), *and I is the number of intersections between the edges of different polygons and is less than* $E \times (E - 1)$.

Proof. The time requirement due to the first three items in the expression $O(E^2 + n \log n + \sum_{i=2}^{n} m_i + k_e E \log E)$ for a single edge is easily seen to be $O(E^3 + M \log M + EM)$. As for the last item $k_e E \log E$, a sum over all the edges leads to $O(IE \log E)$. □

It is very important to point out that, in deriving the upper-bound $O(E^3 + M \log M + EM + IE \log E)$ in Theorem 1, it is assumed that an edge can have $O(E^2)$ critical points and a $Cl(t)$ can have $O(E)$ covering edges. This is however a gross worst case estimate. On average, due to the limited lengths of the edges and L, also depending on the distribution of the edges, the number of critical points on an edge should be much less than the upper-bound $O(E^2)$, so is the size of a $Cl(t)$. Actually, in our experiments presented in Section 6, the total number of critical points on the E edges is found to be more or less in the order of $O(EN^2)$, where N is the number of polygons which is usually much smaller than E. A rigorous analysis on the average case is however needed to verify the above bound, which though is beyond the scope of this paper.

5. Conversion from plane to sphere

It is straightforward to convert the idea of congruent partitioning as described in Sections 3 and 4 from the plane to the spherical domain. When converting to the sphere, a line segment becomes an arc and a circle changes to a small circle on the sphere. An arc whose one end point lies on an edge e of a spherical polygon is now a function $s(t, \theta)$, where t is a real number specifying the point on e and θ is the angle between e and $s(t, \theta)$ on the tangent plane at t. All the analyses of the critical points in the plane can be easily shown to also hold on the sphere. For example, a Type Ia critical point p on an edge e due to another edge u now identifies with the existence of another point q on u such that the arc between p and q is perpendicular to u and has the length equal to the given number L. We however must pay special attention to certain properties unique to the sphere only; this is reflected in the ways on how a characteristic list is constructed and how a critical point is identified.

5.1. Structure of $Cl(p)$

Although the basic structure of a $Cl(p)$ on the sphere is similar to that in the plane, some special treatments need to be introduced due to certain spherical properties. Without loss of generality, suppose the point p is at the north pole (0,0,1). The analogy of the disc of radius L in the plane is a cap on the sphere with its center at p, called the *covering cap* at p, where the small circle that defines the cap is made of those points whose spherical distance to p is L. When $L \geq \pi$, this cap becomes the entire sphere. Let $u_1, u_2, \ldots, u_k$ be the intersections between this cap and the edges of the polygons. The orthogonal projections of these k arcs in the $z = 1$ plane are k partial quadric curves. The $2k$ ordered supporting rays from p to these k curves then form the characteristic list $Cl(p)$. (Notice that each such curve of degree two can have only two supporting rays.) An illustrative example is given in Fig. 12.

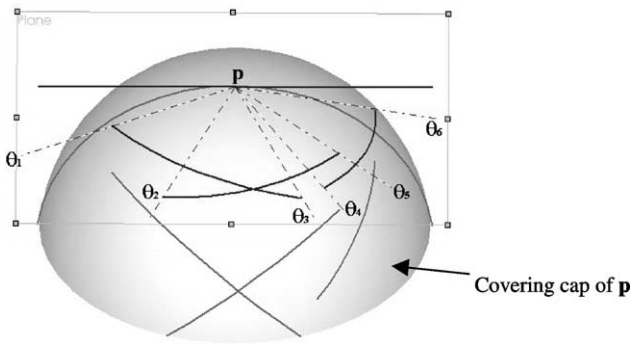


Fig. 12. Characteristic list on the sphere.

5.2. Calculation of critical points

When computing a critical point on the sphere, it is noted that the identification of any of the six types of the critical points is equivalent to finding roots of a system of polynomial equations of degree at most 2 in x, y, and z. As an example, referring to Fig. 13(a), assuming that edge u lies in the $z = 0$ plane, the possible Type Ia critical points that u can contribute to another edge e are determined by the roots of the following system of four equations:

$$x^2 + y^2 = \mathrm{Cos}^2(L)$$

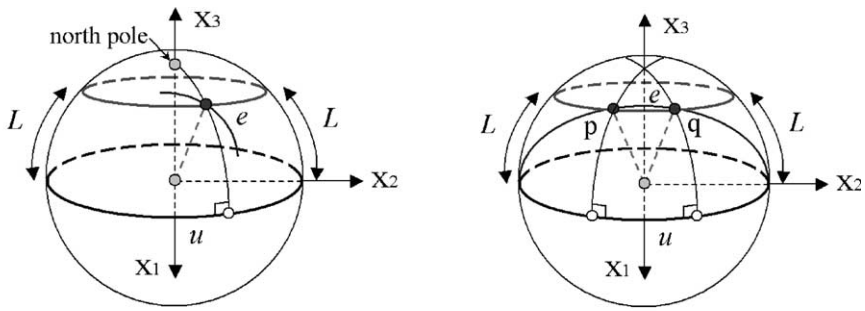
$$z = \pm \mathrm{Sin}(L)$$

$$x^2 + y^2 + z^2 = 1$$

$$ax + by + cz = 0;$$

where L is the given length of the arc, the first two equations define the small circle that is orthogonally L-distance away from u on the sphere, and the last two equations define the edge e with (a, b, c) being the normal vector of the plane of e. It is observed that, due to the non-linearity of the equations, more critical points may emerge. For instance, while in the plane an edge u can contribute at most one Type Ia point to another edge e, two such points could be contributed if it is on the sphere, as demonstrated by Fig. 13(b).

Another special situation, this time a beneficial one, is associated with the length L of the arc. While there is no such parallel analogy in the plane on L, the special number π enjoys a very nice property on the sphere: if $L \geq \pi$, then only Type IIc and Type IId critical points can exist. This assertion directly results from the fact that, when $L \geq \pi$, the covering cap at any point on the sphere is the sphere itself and hence covers all the edges. Since the calculation of a Type IIc or Type IId critical point is independent of L, we



(a) One Type Ia point on the sphere (b) Two Type Ia points on the sphere

Fig. 13. Type Ia points on the sphere.

only need to consider the case of $L < \pi$ when critical points are calculated. Observing that, when $L < \pi$, to an edge e all the arcs $s(t, \theta) : t \in e$ and $\theta \in [0, \pi]$ lie within one hemisphere, we introduce a homogeneous representation for a hemisphere based on the well-known central projection so that the calculation of the critical points can be formulated by the vector algebra, which is found particularly suitable for programming. In Appendix B, a detailed description of this homogeneous representation as well as an example of calculating the Type Ia critical points are provided.

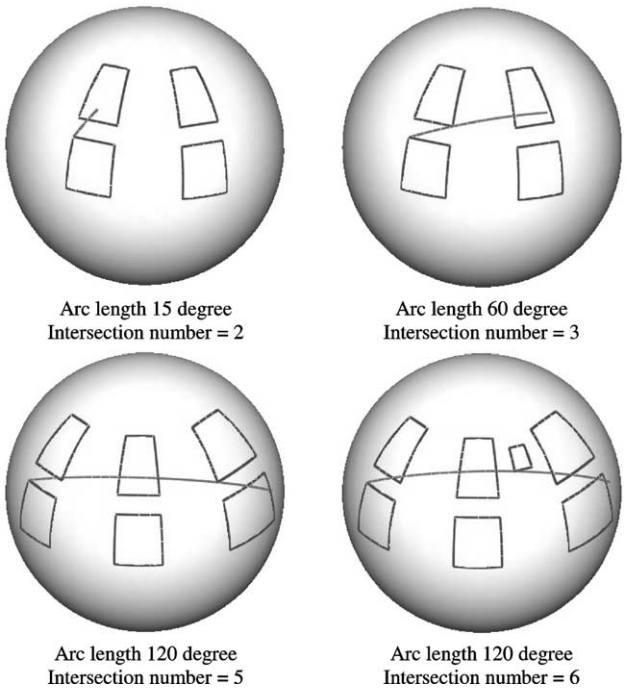


Fig. 14. Test results on some regular patterns (the yellow line is the solution).

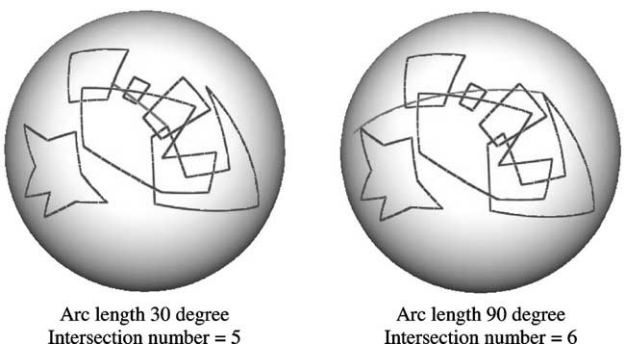


Fig. 15. Test results on an irregular pattern.

6. Implementation and experiments

The proposed algorithm has been implemented on the VC++ platform. In the presence of round-off errors or degenerate cases, the following considerations are taken into account:

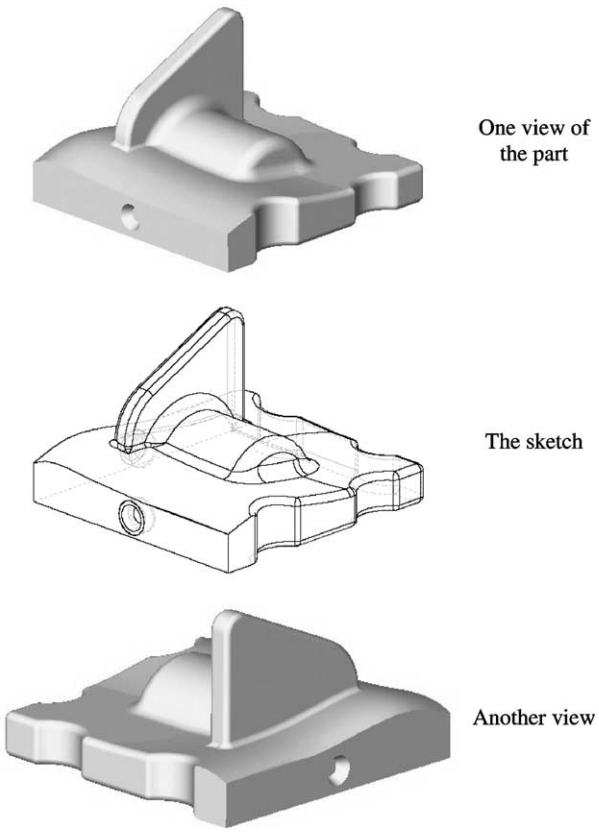


Fig. 16. A mechanical part composed of free-form surfaces and regular patches.

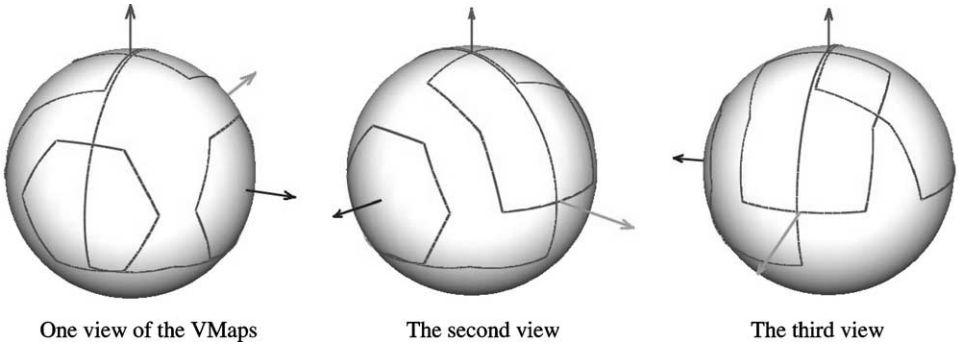


Fig. 17. The simplified approximation of the visibility maps of the free-form part in Fig. 16.

Round-off error and stability. Since the computer stores numbers with a finite precision, round-off error is inevitable and we adopt the standard routines in [12], for solving all the derived formulae to identify critical points. These routines are designed to bound the round-off error accumulation and thus to stabilize the algorithm.

Degeneracies and robustness. In our case degeneracy occurs when several polygons share a common vertex or an edge (cf. spherical polygons in Fig. 20). Since our algorithm is edge-oriented, to deal with these degenerate cases, we preprocess the spherical polygons with line (arc) segment intersection detection to build a doubly connected edge list [5] in the spherical domain. With this data structure, the complete topological information, i.e. the edge-vertex-face relationship is readily obtained. The line segment intersection detection can be performed in an output-sensitive fashion [5]: given n line segments, its time complexity is $O(n \log n + I \log n)$, where I is the number of intersection points of the n segments.

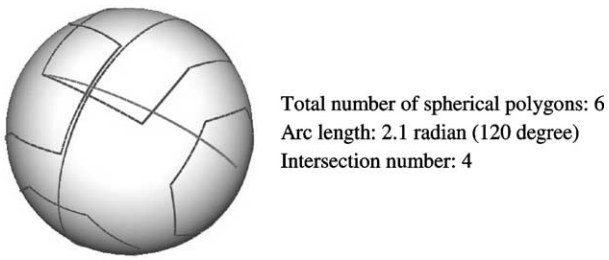


Fig. 18. One solution for the visibility maps shown in Fig. 17.

For demonstration purpose, we first test the algorithm on some simple and artificial patterns on the sphere. Then two tests on some realistic mechanical parts composed of free-form surfaces are performed. In the first experiment, as shown in Fig. 14, tests on some regular patterns with different arc lengths are performed. Next, the algorithm is tested on several irregular patterns with different arc lengths; the results are shown in Fig. 1 and 15.

Finally, we apply the proposed algorithm to two mechanical parts composed of free-form surfaces and regular patches, as displayed in Figs. 16 and 20 (the part surface does not include the bottom flat face). In Fig. 16, the free-form surfaces of the part are grouped into six faces based on some optimization or manufacturing criteria (e.g. certain region on the surface of the part may be required to be cut in a single machining operation), whose (simplified) visibility polygons are depicted in Fig. 17. After applying the proposed algorithm with arc length equal to 2.1 radian

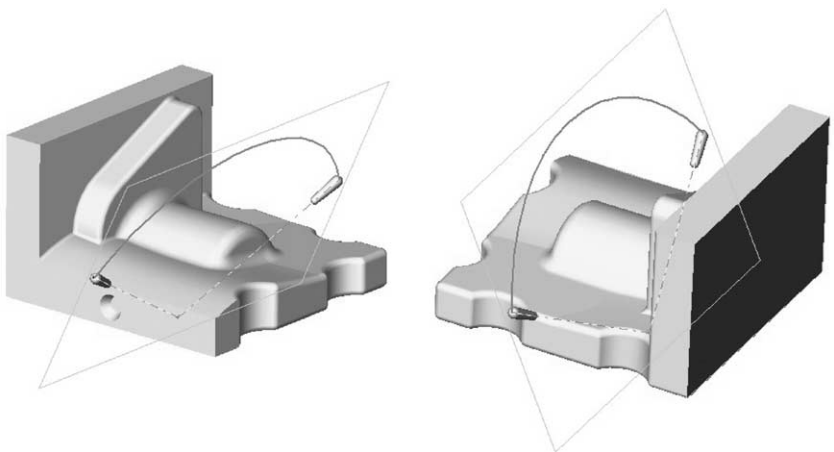


Fig. 19. The semi-finished part after the first setup (the gray color indicates the unmachined surface).

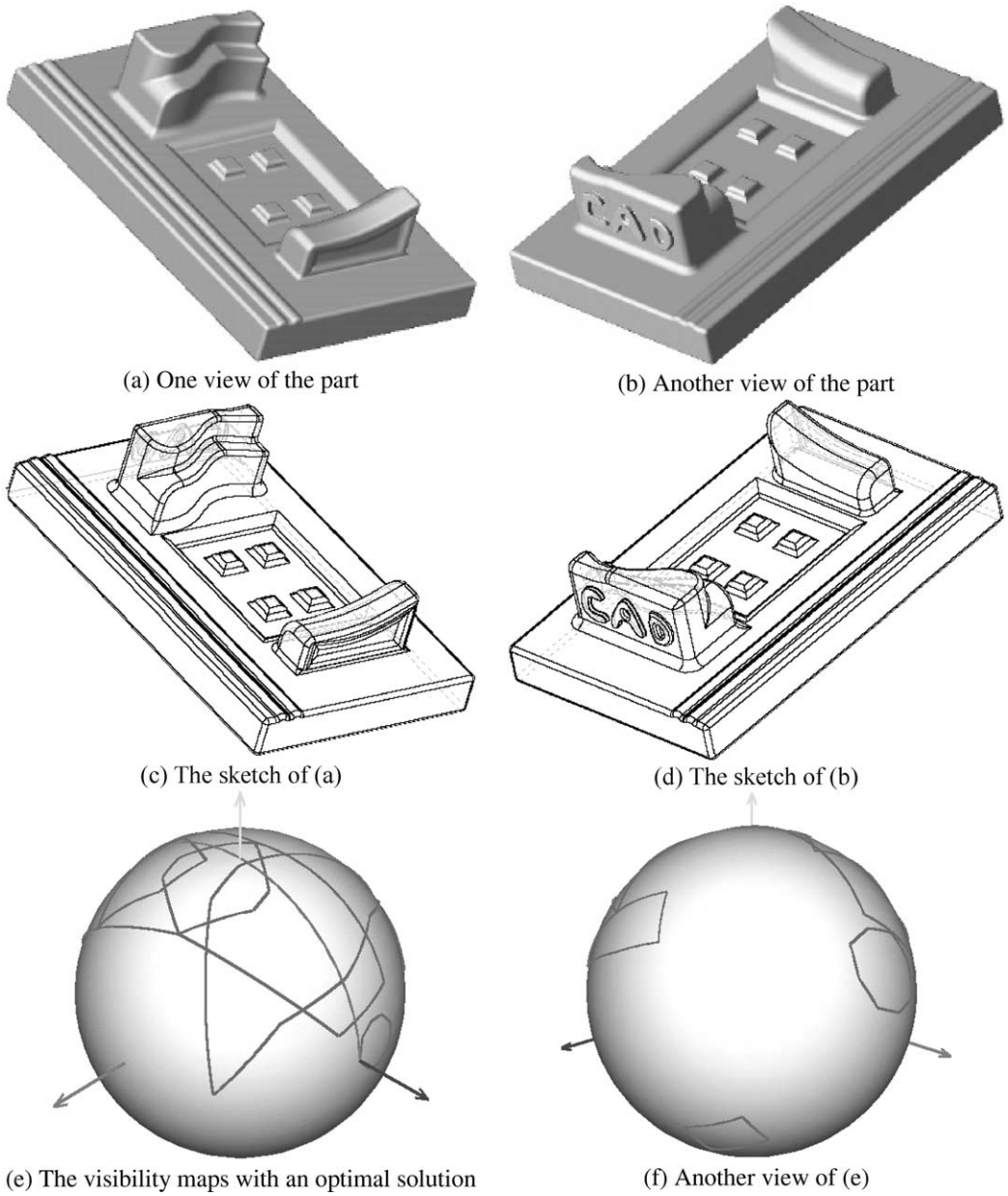


Fig. 20. Another mechanical part and its solution to the maximum intersection problem; the spherical polygons exhibit degeneracies.

Table 1
Running time data for all the tests presented in the paper. The proposed algorithm is implemented using Visual C++ on a PC with a Pentium III 500 MHz processor, 256MB RAM, 12GB hard disk and the operation system is Microsoft Windows 2000

	Figure no. in which the model represented								
	Fig. 1 Poly. No. 6 Ed. No. 41	Fig. 14 a and b Poly. No. 4 Edge. No. 16		Fig. 14c Poly. No. 6 Ed., No. 24	Fig. 14d Poly. No. 7 Ed., No. 28	Fig. 15 Poly. No. 6 Ed., No. 37		Fig. 18 Poly. No. 6 Ed., No. 30	Fig. 20 Poly. No. 8 Ed., No. 51
		a	*b*			*a*	*b*		
Running time (s)	58	4	5	7	7	22	49	25	57

(120 degree), its is found that only four spherical polygons can be intersected at the same time, and thus, at least two setups are needed to machine this part on a 4-axis machine with a θ_r of 120 degree. One representative solution arc is shown in Fig. 18, and the semi-finished part after this setup is displayed in Fig. 19.

We test our program on dozens of spherical cases, of which the running times of all the cases presented in this paper are summarized in Table 1. It should be pointed out however that this table only serves to be a reference; as already mentioned earlier, a rigorous average case analysis and experiments on large collections of spherical polygons are required if an accurate upper bound on the average running time bound is to be established.

7. Conclusion

The primary goal of this paper is to design a deterministic algorithm for solving an important geometric optimization problem: given a set of possibly overlapping spherical polygons on the unit sphere and an arbitrary real number constant $L \leq 2\pi$, find an arc of length L on the unit sphere that intersects as many spherical polygons as possible. To search for a globally optimal solution to this problem, an elaborate congruent partition scheme is proposed by means of exploiting the analytic structure of the solution space with characteristic and eigen lists. Based on the analyses of critical points classification, a simple algorithm with $O(E^4)$ time and $O(E^2)$ space is presented that finds an optimal solution, where E is the total number of edges on the polygons. The proposed algorithm is implemented and experiments with various test patterns are presented for verification purpose.

From theoretical point of view, this paper contributes by offering an exact algorithmic solution to this geometric optimization problem with an arbitrary L—past best results can only handle the two very restrictive special cases of $L = \pi$ and 2π. In terms of practical significance, by allowing the length L to be less than π, the presented algorithm helps find a setup for the workpiece that admits the maximal machining area on a four-axis NC machining whose allowable rotation angular range of the worktable is usually strictly less than 180°.

The algorithm nevertheless has room to improve. First, all the geometric concepts in the paper, such as the characteristic list and congruence relation, are based on individual edges of the polygons but not polygons themselves; the algorithm does not take into account the fact that all the edges on a same polygon will not contribute any critical points to each other. Secondly, it should be further asked whether the fact that our polygons are simple could improve the computational efficiency (e.g. divide a polygon first into its convex components). Last but not the least, we need to run the program on large collections of samples (with more degenerate cases), in order to test its efficiency and robustness.

Appendix A. The proof of order switch in *Cl*(*t*) at a type IIa critical point

A type IIa critical point t is formed when the intersection point p between two line segments u and u' has an L distance from t. Consider the general case that neither u nor u' is perpendicular to the line through p and t.

Without loss of generality, suppose the slope angle α of the line through p and t is greater than $\pi/2$, as shown in Fig. A(a). Suppose the distances from p to t_1 and t_2 are $|p - t_1|$ and $|p - t_2|$, respectively, and suppose the distances from t_1, t_2 to t are ξ and ζ, respectively, with ξ and ζ arbitrarily small. By triangle inequality, it is readily to seen that $\beta < \alpha < \gamma$, $L - \xi < |p - t_1| < L$ and $L < |p - t_2| < L + \zeta$. At t_1, since $L - \xi < |p - t_1| < L$ with an arbitrarily small ξ, the out-angle θ_{uo} of u at t_1 satisfies $\theta_{uo} > \beta$ and the in-angle $\theta_{u'i}$ of u' at t_1 satisfies $\theta_{u'i} < \beta$. Therefore, at t_1 we have

$$\theta_{u'i} < \beta < \theta_{uo}.$$

At t_2, since $L < |p - t_2| < L + \zeta$ with an arbitrarily small ζ, the out-angle φ_{uo} of u at t_2 satisfies $\varphi_{uo} < \gamma$ and the in-angle $\varphi_{u'i}$ of u' at t_1 satisfies $\varphi_{u'i} > \gamma$. Hence, at t_2,

$$\varphi_{u'i} > \gamma > \varphi_{uo}.$$

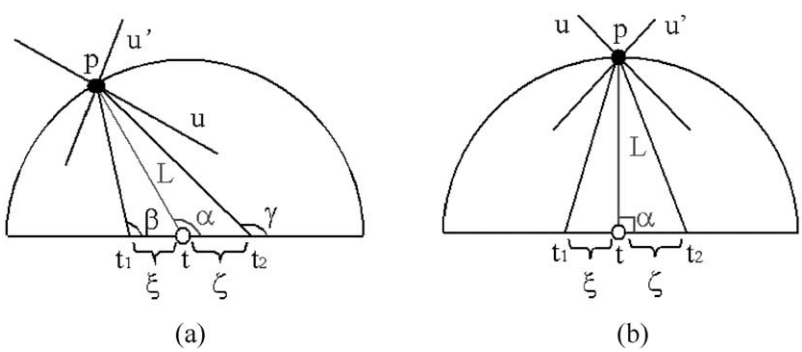


Fig. A.

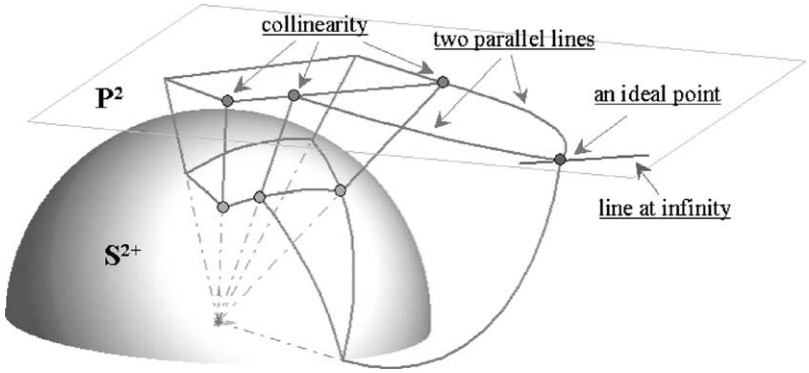


Fig. B. The extended central projection Φ.

Therefore, the order of in- and out-angles of u' and u switches when t is crossed.

The only degenerate case to the above assertion is when the angle α is exactly $\pi/2$, in which the order of the bounding angles in the characteristic list does not change, i.e. the two characteristic lists $Cl(t_1) = \{\theta_{ui}, \theta_{uo}, \theta_{u'i}, \theta_{u'o}\}$ and $Cl(t_2) = \{\varphi_{ui}, \varphi_{uo}, \varphi_{u'i}, \varphi_{u'o}\}$ are equivalent to each other. Meanwhile, the point t becomes an isolated point at which $Cl(t) = \{\eta_{ui}, \eta_{uo} = \theta_{u'i}, \theta_{u'o}\}$. Actually the point t in this case is a false critical point. However, indistinguishing this degeneration from the above general case does not affect the performance of the algorithm.

Appendix B. Homogeneous representation for calculation on $\mathbf{S}^{2+}$

Consider the case with the arc length is less than π. By half space partitioning, up to rotation, only those arc edges lying in the upper hemisphere $\mathbf{S}^{2+}$ (the hemisphere of $\mathbf{S}^2$ corresponding to $x_3 \geq 0$) can possibly contribute critical points to an arc edge e when the θ-domain of arcs $s(t, \theta)$ is limited to $[0, \pi]$. We begin by defining an extension of the central projection [3,4] using a 2D projective plane$\mathbf{P}^2 = \mathbf{R}^3 - (0,0,0)^{\mathrm{T}}$.

Definition 1. Extended central projection $\mathbf{\Phi}$ and its inverse map $\boldsymbol{\xi}$

Denote a point in $\mathbf{S}^{2+}$ by a unit vector $(x_1, x_2, x_3)^{\mathrm{T}}$ in $\mathbf{R}^3$ with the constraints $x_1^2 + x_2^2 + x_3^2 = 1$ and $x_3 \geq 0$. The extended central projection is defined to be the map

$$\mathbf{\Phi}(x_1, x_2, x_3) = (x_1, x_2, x_3) : \mathbf{S}^{2+} \rightarrow \mathbf{P}^2,$$

where the points in $\mathbf{P}^2$ are represented in homogeneous coordinates $\mathbf{x} = (x_1, x_2, x_3)$.The inverse map of $\mathbf{\Phi}$ is defined as

$$\boldsymbol{\xi}(x_1, x_2, x_3) = \mathrm{sign}(x_3)\left(\frac{x_1}{\|\mathbf{x}\|}, \frac{x_2}{\|\mathbf{x}\|}, \frac{x_3}{\|\mathbf{x}\|}\right) : \qquad \mathbf{P}^2 \rightarrow \mathbf{S}^{2+}.$$

Two nice properties of the projection $\mathbf{\Phi}$ are noted without proof [3], as illustrated by Fig. B.

Lemma 5. The projection $\mathbf{\Phi}$ establishes a one-to-one and onto mapping between points in $\mathbf{S}^{2+}$ and points in $\mathbf{P}^2$. In particular, the points on the equator in $\mathbf{S}^{2+}$ are mapped to the ideal points $(x_1, x_2, 0)$ (i.e. points at infinity) in $\mathbf{P}^2$. The projection $\mathbf{\Phi}$ also establishes a one-to-one and onto mapping between great circles in $\mathbf{S}^2$ (except for the equator, semi-circles in $\mathbf{S}^{2+}$) and lines in $\mathbf{P}^2$. In particular, the great circle on the equator in $\mathbf{S}^{2+}$ is mapped to the line at infinity, denoted by homogeneous vector $\mathbf{l}_\infty = (0,0,1)^{\mathrm{T}}$, in $\mathbf{P}^2$.

Lemma 6. The projection $\mathbf{\Phi}$ maps a simple spherical polygon in $\mathbf{S}^{2+}$ to a simple polygon in $\mathbf{P}^2$. Conversely, the inverse of $\mathbf{\Phi}$ maps a simple polygon in $\mathbf{P}^2$ to a simple spherical polygon in $\mathbf{S}^{2+}$.

Since the proposed algorithm is edge-based, only three elementary calculations are required. We formulate these calculations in $\mathbf{P}^2$ with the notion of homogeneous representation, which simplifies the formulae by vector algebra and is particularly suitable for programming. By Lemma 5, the formulae presented below are identical both in the spherical version in $\mathbf{S}^{2+}$ and in the planar version in $\mathbf{P}^2$. In the following, all geometric entities are represented by column vectors, '$\cdot$' and '$\times$' stand for the dot and cross product, respectively.

Calculation rule I. The incidence relationship between points and lines

Since

(1) all the points on the equator $\mathbf{\Gamma}$ in $\mathbf{S}^{2+}$ are collinear on the line at infinity in $\mathbf{P}^2$, and
(2) a line in $\mathbf{S}^{2+} - \mathbf{\Gamma}$ is a semi-circle and any two distinct points $\mathbf{a}, \mathbf{b} \in \mathbf{S}^{2+} - \mathbf{\Gamma}$ uniquely determine a semi-circle,

By Lemma 5, the projection $\mathbf{\Phi}$ is a collineation, i.e. it preserves the collinearity. In $\mathbf{P}^2$, given the homogeneous

representations of points, i.e. $\mathbf{x} = (x_1, x_2, x_3)^{\mathrm{T}}$, and of lines, i.e. $\mathbf{l} = (l_1, l_2, l_3)^{\mathrm{T}}$, we have the following observations:

(1) the point $\mathbf{x}$ lies on the line $\mathbf{l}$ if and only if $\mathbf{x}^{\mathrm{T}}\cdot\mathbf{l} = 0$;
(2) given two lines $\mathbf{l}$ and $\mathbf{l}'$, the intersection point of these two lines is $\mathbf{x} = \mathbf{l}\times\mathbf{l}'$; in particular, two parallel lines, $\mathbf{l} = (l_1, l_2, l_3)^{\mathrm{T}}$ and $\mathbf{l}' = (l_1, l_2, l_3')^{\mathrm{T}}$, are intersected at an *ideal point* (a point at infinity), i.e. $\mathbf{x} = (l_2, -l_1, 0)^{\mathrm{T}}$; and
(3) by duality between the points and lines in $\mathbf{P}^2$, any two distinct points $\mathbf{x}$ and $\mathbf{x}'$ uniquely determine a line $\mathbf{l} = \mathbf{x}\times\mathbf{x}'$; in particular, all ideal points lie on the *line at infinity*, $\mathbf{l}_\infty = (0, 0, 1)^{\mathrm{T}}$.

Calculation rule II. The spherical distance between two points in $\mathbf{P}^2$.

If the space $\mathbf{P}^2$ is associated with the standard Euclidean metric, the projection $\mathbf{\Phi}$ will not preserve the distance, angle and area. Noting that the spherical distance between two points $\mathbf{x}, \mathbf{x}' \in \mathbf{S}^{2+}$ is $\arccos(\mathbf{x}^{\mathrm{T}}\cdot\mathbf{x}') \in [0, \pi]$, we define a distance function in $\mathbf{P}^2$ to be

$$\mathrm{d}(\mathbf{a}, \mathbf{b}) = \arccos(\boldsymbol{\xi}(\mathbf{a})^{\mathrm{T}}\cdot\boldsymbol{\xi}(\mathbf{b})) \in [0, \pi]$$

where $\mathbf{a}$ and $\mathbf{b}$ are two points in $\mathbf{P}^2$. It is trivial to prove that the function $d(\cdot,\cdot) : \mathbf{P}^2 \times \mathbf{P}^2 \to \mathbf{R}$ is a metric and we omit the proof here. Equipped with the metric space $(\mathbf{P}^2, d)$, by Lemma 5, the projection $\mathbf{\Phi} : \mathbf{S}^{2+} \to \boldsymbol{P}^2$ is an isometry, i.e. it preserves the distance.

Calculation rule III. The angle between two lines

We define an angle measure in $\mathbf{P}^2$ such that the angle measured between two lines in $\mathbf{P}^2$ is identical to the spherical angle measured between two semi-circles in $\mathbf{S}^{2+}$. Given two lines $\mathbf{l}$ and $\mathbf{l}'$ in $\mathbf{P}^2$, the angle (measured in radian) between $\mathbf{l}$ and $\mathbf{l}'$ is defined as

$$\alpha_1(\mathbf{l}, \mathbf{l}') = \arccos(\boldsymbol{\xi}(\mathbf{l})^{\mathrm{T}}\cdot\boldsymbol{\xi}(\mathbf{l}')) \in [0, \pi]$$

or

$$\alpha_2(\mathbf{l}, \mathbf{l}') = \pi - \arccos(\boldsymbol{\xi}(\mathbf{l})^{\mathrm{T}}\cdot\boldsymbol{\xi}(\mathbf{l}')) \in [0, \pi]$$

The choice of α_1 or α_2 is dependent on the orientation of lines $\mathbf{l}$ and $\mathbf{l}'$. This identity with the spherical angle in $\mathbf{S}^{2+}$ can be readily proved by the duality between points and lines in $\mathbf{P}^2$.

Finally, we conclude by showing how the three calculation rules are used in calculating a type Ia critical point. To detect a type Ia critical point, one frequently used routine in the proposed algorithm is as follows. Given a line $\mathbf{l}$ determined by two distinct points $\mathbf{a}, \mathbf{b} \in \mathbf{P}^2$ and a third point $\mathbf{p} \in \mathbf{P}^2$ not lying in $\mathbf{l}$,

(1) find the point $\mathbf{q}$ lying in $\mathbf{l}$ such that the line $\mathbf{l}'$ determined by $\mathbf{p}$ and $\mathbf{q}$ is perpendicular to $\mathbf{l}$, and
(2) find the spherical distance between point $\mathbf{p}$ and line $\mathbf{l}$.

By rule I, $\mathbf{l} = \mathbf{a}\times\mathbf{b}$, $\mathbf{l}' = \mathbf{p}\times\mathbf{q}$, and $\mathbf{q}^{\mathrm{T}}\cdot\mathbf{l} = 0$. Since $\mathbf{l}$ and $\mathbf{l}'$ are perpendicular to each other, by rule III, $\mathbf{l}^{\mathrm{T}}\cdot\mathbf{l}' = 0 \Rightarrow (\mathbf{a}\times\mathbf{b})^{\mathrm{T}}\cdot(\mathbf{p}\times\mathbf{q}) = 0$. Then $\mathbf{q}$ is the solution of the linear equation system

$$\begin{cases} (\mathbf{a}\times\mathbf{b})^{\mathrm{T}}\cdot\mathbf{q} = 0 \\ (\mathbf{a}\times\mathbf{b})^{\mathrm{T}}\cdot(\mathbf{p}\times\mathbf{q}) = 0 \end{cases} \Rightarrow \mathbf{q} = (\mathbf{a}\times\mathbf{b})\times((\mathbf{a}\times\mathbf{b})\times\mathbf{p})$$

The last identity is obtained by using the scalar triple product

$$[\mathbf{A}, \mathbf{B}, \mathbf{C}] = \mathbf{A}^{\mathrm{T}}\cdot(\mathbf{B}\times\mathbf{C}) = \mathbf{B}^{\mathrm{T}}\cdot(\mathbf{C}\times\mathbf{A}) = \mathbf{C}^{\mathrm{T}}\times(\mathbf{A}\times\mathbf{B})$$

Finally, by rule II, the spherical distance between $\mathbf{p}$ and line $\mathbf{l}$ is

$$d(\mathbf{p}, \mathbf{q}) = \arccos(\boldsymbol{\xi}(\mathbf{p})^{\mathrm{T}}\cdot\boldsymbol{\xi}(\mathbf{q})) \in [0, \pi].$$

References

[1] Brown KQ. Geometric transformation for fast geometric algorithms. PhD Dissertation. Department of Computer Science, Carnegie Mellon University; 1979.
[2] Chazelle B, Guibas LJ, Lee DT. The power of geometric duality. BIT 1985;25:76–90.
[3] Chen LL, Woo T. Computational geometry on the sphere with applications to automated machining. Trans ASME J Mech Des 1992; 114:288–95.
[4] DasGupta B, Roychowdhury VP. In: Du D-Z, Sun J, editors. Two geometric optimization problems. New advances in optimization and approximation, Dordecht: Kluwer; 1994. p. 30–57.
[5] de Berg M, van Kreveld M, Overmars M, Schwarzkopf O. Computational geometry. Berlin: Springer; 1998.
[6] do Carmo MP. Differential geometry for curves and surfaces. Englewood Cliffs, NJ: Prentice-Hall; 1976.
[7] Fomenko AT, Kunii TL. Topological modeling for visualization. Berlin: Springer; 1997.
[8] Gan JG. Spherical algorithms for setup orientations of workpieces with sculptured surfaces. PhD Dissertation. Ann Arbor, MI: Department of Industrial and Operations Engineering, University of Michigan; 1990.
[9] Gan JG, Woo TC, Tang K. Spherical maps: their construction, properties, and approximation, Trans. ASME J Mech Des 1994;116: 357–63.
[10] Garey MR, Johnson DS. Computers and intractability. San Francisco: Freeman; 1979.
[11] Gupta P, Janardan R, Majhi J, Woo T. Efficient geometric algorithms for workpiece orientation in 4- and 5-axis NC machining. Comput-Aided Des 1996;28(8):577–87.
[12] Press WH, Teukolsky SA, Vetterling WT, Flannery BP. Numerical recipes in C + +: the art of scientific computing, 2nd ed. Cambridge: Cambridge University Press; 2002.
[13] Tang K, Woo T, Gan J. Maximum intersection of spherical polygons and workpiece orientation for 4- and 5-axis machining. Trans ASME J Mech Des 1992;114:477–85.
[14] Tang K, Chen LL, Chou SY. Optimal workpiece setups for 4-axis numerical control machining based on machinability. Comput Indus 1998;37:27–41.

Kai Tang is currently a faculty member in the Department of Mechanical Engineering at Hong Kong University of Science and Technology. Before joining HKUST in 2001, he had worked many years in the CAD/CAM and IT industries. His research interests concentrate on designing efficient and practical algorithms for solving real world computational, geometric, and numerical problems. Dr Tang received PhD in Computer Engineering from the University of Michigan in 1990, MSc in Information and Control Engineering in 1986 also from the University of Michigan, and BSc in Mechanical Engineering from Nanjing Institute of Technology in China in 1982.

Yong-Jin Liu is currently a PhD student at Hong Kong University of Science and Technology (HKUST). He enrolled Tianjin University in 1994, exempted from entrance examination by receiving national high school student mathematics and physics competition awards. After receiving his B.Eng. in Mechano-Electronic Engineering, he enrolled HKUST in 1998, exempted from entrance examination with recommendation of the Ministry of Education, P.R. China. He received his MPhil degree in 1999 in Mechanical Engineering at HKUST. His research interests include geometric modeling, design automation and optimization, computer graphics.