Optimal algorithms for some proximity problems on the Gaussian sphere with applications.

Prosenjit Gupta* Sanjeev Saluja*

September 4, 1996

Abstract

We consider some geometric problems on the unit sphere which arise in NC-machining. Optimal linear time algorithms are given for these problems using linear and quadratic programming in three dimensions.

Keywords: NC-machining, Gaussian sphere, computational geometry, linear programming.

1 Introduction

On a numerically controlled (NC) machine, the cutting tool is controlled by a program, containing instructions of workpiece setups and tool paths for the machining of a component. NC-machines are typically classified as 3-, 4-, or 5-axis machines, depending on the number of degrees of freedom enjoyed by the cutter. The ability of a cutter to access a workpiece surface depends not only on the degrees of freedom but also on the nature of the cutter itself. Cutters are classified as flat-end, fillet-end, and ball-end cutters, depending on the maximum angle, θ, that the cutter's axis can be tilted from the normal to the surface at a given point. In a *flat-end* cutter, $\theta = 0°$; in a *fillet-end*

*Max-Planck-Institut für Informatik, Im Stadtwald, D 66123 Saarbrücken, Germany. Email:{pgupta,saluja}@mpi-sb.mpg.de

cutter, $\theta < 90°$; and in a *ball-end* cutter, $\theta = 90°$. Cutter selection is often determined by the range of surface normals, the requirements of surface texture, and the demands on dimensional accuracy [1].

In this paper, we consider some algorithmic problems on the unit sphere that arise in NC-machining: checking if a point set on the unit sphere can be enclosed within a hemisphere, finding the smallest enclosing circle for a hemispherical point set on the unit sphere and finding the largest empty circle lying inside a given *spherical* polygon in a hemisphere on the surface of the unit sphere. For the first problem an $O(n \log n)$ solution was suggested by Chen and Woo [2]. For the next two problems, Gan et al. [4] have given $O(n \log n)$ algorithms. We show how to improve these algorithms to $O(n)$ using linear and quadratic programming in three dimensions.

We briefly review the basic ideas and results of linear and quadratic programming that we shall use in this paper. A linear program in $\mathbb{R}^3$ consists of

1. Real variables x_1, x_2, x_3.
2. Linear constraints over the variables:
   $a_{i1}x_1 + a_{i2}x_2 + a_{i3}x_3 \geq \beta_i, i = 1 \cdots n$ where the $a_{ij}s$ and β_is are real numbers.
3. Linear minimization function (also called the *objective function* of the program:
   $\gamma_1 x_1 + \gamma_2 x_2 + \gamma_3 x_3$ where $\gamma_i's$ are real numbers.

A *feasible solution* of the above linear program is an assignment of real values to x_1, x_2, x_3 which satisfies all the constraints in (ii). An *optimal solution* to the program is a feasible solution which gives gives the smallest possible value to the objective function among all possible feasible solutions. The set of all possible feasible solution is called the feasible region of the program.

The version of quadratic programming in $\mathbb{R}^3$ that we consider consists of:

1. Real variables x_1, x_2, x_3.
2. Linear constraints over the variables:
   $a_{i1}x_1 + a_{i2}x_2 + a_{i3}x_3 \geq \beta_i, i = 1 \cdots n$ where the $a_{ij}s$ and β_is are real numbers.
3. Quadratic minimization function (i.e. the *objective function* of the program:
   $\gamma_1 x_1^2 + \gamma_2 x_2^2 + \gamma_3 x_3^2 + z^2$ where $\gamma_i's$ are real numbers; $\gamma_3^2 \leq 4\gamma_1\gamma_2$; $\gamma_1, \gamma_2 \geq 0$;

The feasible and optimal solutions of the quadratic program are defined analogous to those of linear program.

Megiddo [7] has shown that there algorithms which solve the linear and quadratic program in 3 dimensions in $O(n)$ steps where n is the number of constraints. Henceforth, we refer to these results as the following theorems.

Theorem 1.1 *[7] There is an algorithm with worst case time-complexity $O(n)$ to compute an optimal solution to linear program in $\mathbb{R}^3$ with n constraints. Hence there is also an $O(n)$ algorithm to compute a feasible solution of a set of n linear constraints in 3 variables.*

Theorem 1.2 *[7] There is an $O(n)$ algorithm to compute an optimal solution to a quadratic program in $\mathbb{R}^3$ with n constraints.*

For more details on linear and quadratic programming, we refer the reader to [3][5].

2 Preliminaries

Formally, we define the *unit sphere* as $\mathbb{S}^2 = \{P \in \mathbb{R}^3 | \ |P| = 1\}$. A point P in $\mathbb{S}^2$ is a unit vector in $\mathbb{R}^3$ and is represented by the three-tuple (P_1, P_2, P_3). Two points P and Q of $\mathbb{S}^2$ are *antipodal* if $P = -Q$. For point $P \in \mathbb{S}^2$, we define a *great circle* $G(P)$ with pole P as $G(P) = \{x \in \mathbb{S}^2 | \ P \cdot x = 0\}$. Note that $G(P) = G(-P)$. All regions on the surface of the sphere are treated as point sets and union ($\cup$), intersection ($\cap$) and set inclusion ($\in$) and containment ($\subseteq$) have their usual meanings. We use $int(\cdot)$ and $bd(\cdot)$ to denote the interior and boundary of a region respectively. We define the *distance*, $d(P, Q)$, between two points P and Q of $\mathbb{S}^2$ as the length of the shorter of the two great arcs joining P and Q, i.e., $d(P, Q) = \cos^{-1}(P \cdot Q)$. A *spherical disk* $D(P, r)$ with pole P and radius r, where $0 \leq r \leq \pi/2$ is defined as $D(P, r) = \{X \in \mathbb{S}^2 | \ d(P, X) \leq r\}$. Throughout, we shall refer to a spherical disk simply as a *disk*. Note that the circle $bd(D(P, r))$ is a great circle if $r = \pi/2$; otherwise it is a *small circle*. Also note that $D(P, \pi/2)$ is a hemisphere with pole P.

Given any two distinct points P and Q in $\mathbb{S}^2$, which are not antipodal, there is a unique great circle passing through P and Q, defined as $circ(P, Q) = G((P \times Q)/|P \times Q|)$. We define the shorter of the two arcs of $circ(P, Q)$ joining P and Q as *segment* $\overline{PQ}$. A *spherical polygon* $\mathcal{P}$ on $\mathbb{S}^2$ (or *polygon*, for short) is defined as the closed region bounded by a path of segments $\overline{P_1P_2}$, $\overline{P_2P_3}$, ..., $\overline{P_nP_1}$, where P_i, $i = 1, \ldots, n$, are the vertices of $\mathcal{P}$.

The Gaussian map (or GMap) of a surface is a map of normals to the surface on the unit sphere. The *visibility map* (or VMap, for short) of a surface is the set of all directions along which an unobstructed line-of-sight can be established from the cutter (which is treated as a point) to every point on the surface. For a ball-end cutter, the VMap of a surface can be represented on the unit sphere $\mathbb{S}^2$ as a *spherical polygon*, i.e., a closed region on $\mathbb{S}^2$ bounded by arcs of great circles.

3 The hemisphericity test

In this section, we consider the problem of detecting if a set S of points in $\mathbb{S}^2$ can be enclosed in a hemisphere and reporting such an

hemisphere if it exists. Detection of hemisphericity is central to several algorithms for problems on spheres [2]. An $O(n \log n)$ time algorithm for this problem is given by Chen and Woo [2]. In this section we suggest a simpler and faster algorithm for the problem. We show how to reduce the problem to an instance of linear programming in $\mathbb{R}^3$ which can be solved in optimal $\Theta(n)$ time, using the algorithm of Megiddo [7].

Lemma 3.1 *A set S of n points on $\mathbb{S}^2$ is hemispherical iff $\mathcal{R}(S) = \cap_{p \in S} D(p, \pi/2) \neq \emptyset$.*

Proof:
$\Longrightarrow$ Let S be hemispherical. Let x be the pole of the hemisphere containing S. For any $p \in S$, $p \in D(x, \pi/2)$. Hence $x \in D(p, \pi/2)$. Therefore $x \in \cap_{p \in S} D(p, \pi/2)$ i.e. $\mathcal{R}(S) \neq \emptyset$.

$\Longleftarrow$ If $\mathcal{R}(S) = \cap_{p \in S} D(p, \pi/2) \neq \emptyset$, then let x be a point in $\cap_{p \in S} D(p, \pi/2)$. Hence for all p in S, $p \in D(x, \pi/2)$, i.e. S is hemispherical. ■

Given a set S of n points in $\mathbb{S}^2$, we construct a linear program $\mathcal{P}(S)$ in variables X, Y, Z as follows: for every point $p_i = (x_i, y_i, z_i) \in S$, define constraint C_i as $x_i X + y_i Y + z_i Z \geq 0$. Note that constraint $C_i's$ express the fact that the dot-product between any given point p_i and a feasible solution point $P = (X, Y, Z)$ is nonnegative i.e. $d(P, p_i) \leq \pi/2 \quad (i = 1, ..., n)$. This immediately implies the following.

Lemma 3.2 *Let $F(S)$ be the feasible region of the linear program $\mathcal{P}(S)$. Then $F(S) \neq \emptyset$ iff $\mathcal{R}(S) = \cap_{p \in S} D(p, \pi/2) \neq \emptyset$.* ■

Lemma 3.3 . *S is hemispherical if and only if the linear program $\mathcal{P}(S)$ has a feasible solution.*

Proof: Follows from Lemmas 3.1 and 3.2. ■

We now give the steps of algorithm $HEMI$ to check if a given set S of points in $\mathbb{S}^2$ lies in a hemisphere.

HEMI

1. For each point $p_i \in S$ create the constraint C_i as above.
2. Use the algorithm of Megiddo (Theorem 1.1) to find a feasible solution to the set of constraints C_is.

Theorem 3.1 *Given a set S of n points on $\mathbb{S}^2$, algorithm HEMI determines in $\theta(n)$ time if S can be enclosed in a hemisphere.*

Proof: Note that Step 1 takes $O(1)$ time to create the constraint for each point and hence $O(n)$ time to create the n constraints. Further it follows from Theorem 1.1 that Step 2 to find a feasible solution to n constraints. Therefore the running time of HEMI on an input of n points is $O(n)$. ■

4 Computing the smallest enclosing circle for points in a hemisphere

The VMap represents a set of feasible directions for cutter orientation. Since a disk contained in the VMap represents a subset of such directions, we can approximate a VMap by the largest empty circle contained inside it. Gan et al. [4] have shown that if the largest empty disk inside a VMap is $D(p,r)$, then the smallest disk containing the corresponding GMap is $D(p,\pi/2-r)$. Thus to approximate the VMap, it is sufficient to construct the smallest disk containing the corresponding GMap. This motivates the following problem: Given a set S of n points in a hemisphere on $\mathbb{S}^2$, find the smallest disk containing S. In [4], Gan et al. have given an $O(n \log n)$ algorithm for this problem. In this section, we give an algorithm which has a worst case running time of $O(n)$, which is asymptotically optimal. The algorithm first determines the hemisphere containing the points (assuming it is not known in advance) and determines if the smallest enclosing circle is the hemisphere itself; if not then the problem is transformed into a minimization problem over three variables with n linear constraints and a quadratic objective function. Next this minimization problem is solved using the algorithm of Megiddo (Theorem 1.2) which runs in worst case time which is linear in the number of constraints.

We discuss now the first step of the algorithm. Let $p_1, ..., p_n$ be the given set of points which lie in a hemisphere on $\mathbb{S}^2$. Note that since the points lie in a hemisphere, the smallest enclosing circle C is defined by a plane H which cuts the sphere in C and H is the farthest plane from origin which separates origin from every point in the given point set (Figure 1). We distinguish two cases depending on whether H contains the origin or not. If H contains the origin, then either three of the points lie on the great circle bounding the hemisphere (Figure 3) or two of the points lie diagonally opposite on the great circle bounding the hemisphere (Figure 2). Either of the two conditions can be detected easily, if we know the hemisphere containing the points. In case, the hemisphere which contains the given point set is not known already, we can determine it in $O(n)$ time by using the algorithm HEMI of Section 3.

If H does not contain the origin then we can reduce the given point set to a minimization problem as follows. Let $\langle x_i, y_i, z_i \rangle$ be the coordinates of point p_i and $ax + by + cz = 1$ be the equation of an arbitrary plane (not containing origin) in three dimensions. The distance of this plane from origin is given by $\frac{1}{\sqrt{a^2+b^2+c^2}}$. Further, maximizing the distance of the plane from the origin can be expressed equivalently as minimizing the quadratic convex objective function $a^2 + b^2 + c^2$. We can reduce the problem to the solution of the following optimization problem $(\mathcal{P})$ in variables a, b, c:
Minimize $a^2 + b^2 + c^2$ w.r.t.
$ax_i + by_i + cz_i \geq 1, \quad 1 \leq i \leq n$

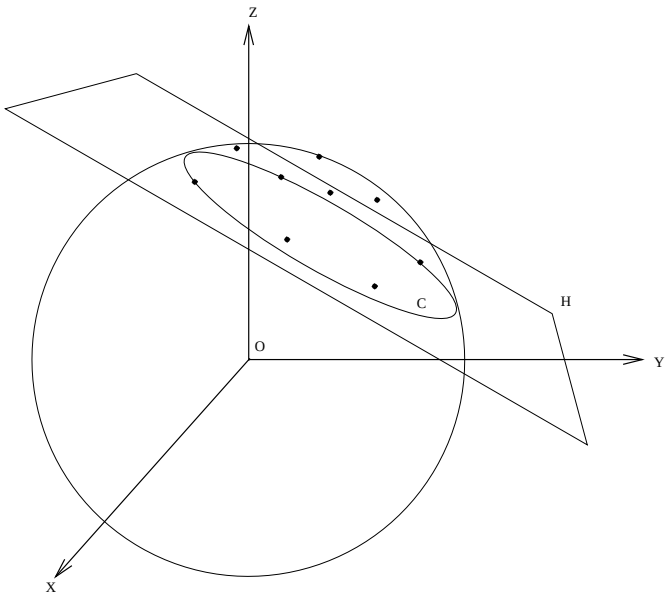

Figure 1: *Plane defining smallest enclosing circle*

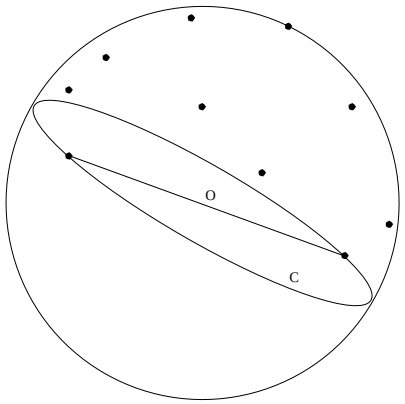

Figure 2: *Two points on bounding great circle*

Note that the direction of the inequalities ensures that all the given points lie on that side of the plane $ax + by + cz = 1$ which does not contain the origin. So any feasible solution to the above set of constraints corresponding to a plane separating origin from the given set of points.

We can now summarize the algorithm SEC for the smallest enclosing circle problem as follows :

SEC

1. Use the algorithm **HEMI** of the Section 3 to find the hemisphere which contains the given point set.
2. Check if three points lie on the periphery of the hemisphere or two points lie diametrically opposite on the periphery. This can be done by determining (in $O(n)$ time) which inequalities in the solution of the linear program in HEMI are satisfied exactly by the optimum solution. If these points are found, then determine the great circle defining the periphery of the hemisphere is the required circle and stop. Output this great circle and Stop.
3. If the condition checked in Step 2 does not hold, then solve the optimization problem $\mathcal{P}$ to obtain the coefficients of the plane defining the smallest enclosing circle. Determine and output the corresponding circle and Stop.

Theorem 4.1 *Given a set S of n hemispherical points on $\mathbb{S}^2$, the smallest disk containing all points in S can be constructed in $O(n)$ time.*

Proof: Each of the three steps has a worst case time complexity of $O(n)$. Therefore, the worst case time complexity of the algorithm SEC is $O(n)$. ■

Corollary 4.1 *Given a spherical polygon $\mathcal{P}$ inside a hemisphere on $\mathbb{S}^2$ with n vertices, the largest disk contained in $\mathcal{P}$ can be found in $O(n)$ time.*

Proof: In [4], it is shown that finding the largest disk enclosed in a spherical polygon with n vertices in a hemisphere is equivalent to finding the smallest enclosing circle of a set of n points which lie in a hemisphere. Therefore the result follows from Theorem 4.1. ■

References

[1] L. Chen, S. Chou, and T. Woo. Separating and intersecting spherical polygons: computing machinability on three-, four-, and five-axis numerically controlled machines. *ACM Transactions on Graphics*, 12:4:305–326, 1993.

[2] L.L. Chen and T. Woo. Computational geometry on the sphere with applications to automated machining. *Journal of Mechanical Design*, 114:288–295, 1992.

[3] V. Chvátal. Linear programming. *Freeman Publishers*, 1983.

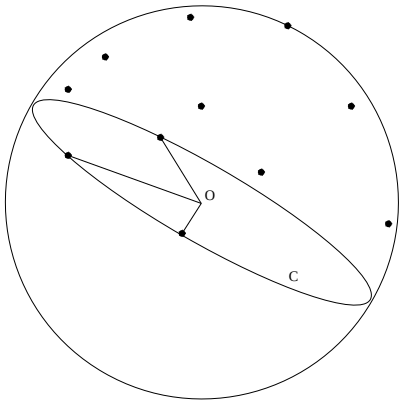

Figure 3: *Three points on bounding great circle*

[4] J.G. Gan, T.C. Woo, and K. Tang. Spherical maps: their construction, properties, and approximation. *Journal of Mechanical Design*, 116:357–363, 1994.

[5] Martin Grötschel, László Lovász and Alexander Schrijver. Geometric algorithms and combinatorial optimization. *Springer Verlag Publishers*, 1988.

[6] F.P. Preparata and M.I. Shamos. *Computational Geometry – an introduction.* Springer–Verlag, 1988.

[7] N. Megiddo. Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems. *SIAM Journal on Computing*, 12:4:759–776, 1983.

[8] K. Tang, T. Woo, and J. Gan. Maximum intersection of spherical polygons and workpiece orientation for 4- and 5-axis machining. *Journal of Mechanical Design*, 114:477–485, 1992.