On Removing Extrinsic Degeneracies in Computational Geometry*

Francisco Gómez
Department of Applied Mathematics
Universidad Politecnica de Madrid
Madrid, Spain

Suneeta Ramaswami
Department of Computer Science
Rutgers University
Camden, NJ, USA

Godfried Toussaint
School of Computer Science
McGill University
Montréal, Québec, Canada

Abstract

Existing methods for removing degeneracies in computational geometry can be classified as either *approximation* or *perturbation* methods. These methods give the implementer two rather unsatisfactory choices: find an *approximate* solution to the *original* problem given, or find an *exact* solution to an *approximation* of the original problem. We address an alternative approach for removing *extrinsic* degeneracies that has received little attention in the computational geometry literature. Often a typical computational geometry paper will make a non-degeneracy assumption that can in fact be removed (without perturbing the input) by a global rigid transformation of the input. Once the solution is

*This research was carried out during the first author's visit to and the second author's post-doctoral fellowship at McGill University in 1995. The first author was self-supported, and the second and third authors were supported by NSERC Grant no. OGP0009293 and FCAR Grant no. 93-ER-0291.

obtained on the transformed non-degenerate input, it can be transformed back trivially to yield the solution to the original problem. In these situations, by applying suitable *pre-* and *post-* processing steps to an algorithm, we obtain the *exact* solution to the *original* problem using the algorithm that assumes a non-degenerate input, even when that input is in fact degenerate.

We consider several non-degeneracy assumptions that are typically made in the literature, propose algorithms for performing the pre- and post- processing steps that remove these degeneracies, analyze their complexity and, for some of these problems, give lower bounds on their worst-case complexity. The assumptions considered here include: (1) no two points in the plane have the same x-coordinate, (2) no two points in space lie on a vertical line, (3) no two points in space have the same x-coordinate, (4) no three points in space lie on a vertical plane, and (5) no two line segments lie on a vertical plane.

Incorporating our algorithms with those in the literature that make these non-degeneracy assumptions, allows those algorithms to work even when the degeneracies are present, albeit at the cost of increased complexity.

We propose low-degree polynomial-time solutions for the decision, computation and optimization versions of all these problems. For the optimization version of problem (5) we give an $O(n^4)$ time algorithm, reducing the previous best running time of $O(n^6 \log n)$.

1 Introduction

Algorithms in computational geometry are usually designed for the real RAM (random access machine) assuming that the input is in *general position.* More specifically, the general position assumption implies that the input to an algorithm for solving a specific problem is free of certain degeneracies. Yap [34] has distinguished between intrinsic or *problem-induced* and *algorithm-induced* degeneracies. For example, in computing the convex hull of a set of points in the plane, where "left" turns and "right" turns are fundamental primitives, three colinear points constitute a problem-induced degeneracy. On the other hand, for certain vertical line-sweep algorithms two points with the same x-coordinate constitute an algorithm-induced degeneracy. Computational geometers make these assumptions because doing so makes it not only much easier to design algorithms but often yields algorithms with reduced worst-case complexities (see [7], for example). On the other hand, to the implementers of geometric algorithms these assumptions are frustrating. Programmers would like the algorithms to work for any input that they may encounter in practice, regardless of the degeneracies such an input contains. In this paper, we discuss techniques to remove some commonly occuring algorithm-induced, or *extrinsic*, degeneracies in the input set.

Due to the practical importance of having algorithms work correctly for degenerate input, there has recently been a flurry of activity on this problem in the computational geometry literature. A survey of various techniques available for coping with degeneracies, and their interaction with the real RAM assumption, was written by Fortune [22]. In one class of methods to deal with degeneracies for solving a problem, the approach is to compute some *approximation* to the exact solution of the problem when a degenerate input is encountered. For example, Iri and Sugihara [25] proposed an algorithm for constructing the planar Voronoi diagram that does not "crash" when it encounters certain degeneracies. It does this by computing a topologically consistent output that may, however, be numerically incorrect. The other well known class of methods for handling degeneracies is via *perturbation*. Here the input is perturbed in some way by an infinitesimal amount so that the degeneracies are no longer present in the perturbed input. Problem-dependent perturbation methods have actually been around for some time. For example in 1963 Dantzig [12] proposed one such scheme for linear programming. For a review of other examples of problem-specific perturbation schemes, see [20, 32]. More general and powerful methods have recently been proposed by Edelsbrunner and Mücke [17], Yap [34, 33] and Emiris and Canny [19, 20]. An insightful discussion of perturbation methods and their short-comings is given by Seidel [32].

The above methods give the implementer two rather unsatisfactory choices: find an *approximate* solution to the *original* problem given, or find an *exact* solution to an *approximation* of the original problem. Sometimes it may be possible to convert the approximate solution obtained from perturbation methods to the exact solution by using some kind of *post-processing* step but this step may be difficult and complicated [9]. In fact, Seidel [32] questions the wisdom of using perturbation methods altogether.

In this paper we address an issue concerning the assumption of extrinsic non-degeneracies which has received little attention in the computational geometry literature. Often a typical computational geometry paper will make a non-degeneracy assumption that can in fact be removed (without perturbing the input) by a global rigid transformation of the input (such as a rotation, for example). Once the solution is obtained on the transformed non-degenerate input, the solution can be transformed back trivially (by an inverse rotation) to yield the solution to the original problem. In these situations, by applying suitable *pre-* and *post-* processing steps, we obtain the *exact* solution to the *original* problem using an algorithm that assumes a non-degenerate input, even when that input is in fact degenerate.

We consider several extrinsic non-degeneracy assumptions that are typically made in the literature, propose efficient algorithms for performing the pre- and post-processing steps (suitable rotations) that remove these degeneracies, analyze their complexity in the real RAM model of computation and, for some of these problems, give lower bounds on their worst-case complexity. While degeneracies are a practical concern and the real RAM may be, from the practical standpoint, a poor model if numbers become long due to computation of rotations, we restrict ourselves in this work to infinite precision in order to focus on the purely geometric aspects of removing algorithm-induced degeneracies. In the practical implementation of our algorithms, where numerical robustness is important as well, the rational rotation methods of Canny, Donald and Ressler [10] may be incorporated.

The assumptions considered here include:

1. no two points in the plane have the same x-coordinate,
2. no two points in space lie on a vertical line,
3. no two points in space have the same x-coordinate,
4. no three points in space lie on a vertical plane,
5. no two line segments lie on a vertical plane

Therefore, incorporating our algorithms with those in the literature that make these non-degeneracy assumptions allows those algorithms to work even when the degeneracies are present, albeit at the cost of increased complexity.

These problems are also intimately related to those of obtaining "nice" orthogonal projections of polygons and skeletons of spatial subdivisions in such disciplines as knot-theory [27], graph-drawing [6] and visualization in computer graphics [5]. It is worth pointing out that in the visualization community, the above situations ((1)-(5)) are sometimes referred to simply as degeneracies in the input (see [26], for example), without necessarily making the distinction between intrinsic and extrinsic. Removing assumptions (1), (2) and (3) is equivalent to computing so-called *regular* and *Wirtinger* projections of point sets. Removing assumption (4) is equivalent to finding a plane such that the orthogonal projection of the points onto that plane has no three projected points co-linear. Removing assumption (5) ensures that no two projected edges are co-linear and is a well known projection method for visualizing "wire-frame" objects [26]. We would like to point out that the techniques presented in this paper may be used to find projections (of points in space) that contain no four co-circular points. However, this method would require an oracle that computes intersections between high-degree curves on a sphere. Since such a model of computation is impractical, we do not present the result here. Also, in order to keep the paper self-contained, we will discuss new techniques as well as results that follow from existing methods in computational geometry.

Looking at these degeneracy problems from the point of view of finding "nice" projections, uncovers additional desirable properties that we would like the transformed non-degenerate input to have. In order to increase the *robustness* of our solution even further, we are interested in obtaining the *most* non-degenerate input possible. Hence the five problems itemized above also have their optimization counterparts. We also give algorithms for such *optimal* removal of degeneracies.

We propose low-degree polynomial-time solutions for the decision, computation and optimization versions of all these problems. For the optimization version of problem (5) we give an $O(n^4)$ time algorithm, reducing the previous best running time of $O(n^6 \log n)$ [26].

2 No Two Planar Points on a Vertical Line

Many papers in computational geometry assume that no two points of a given planar set S have the same x-coordinate. For example, in [29] the assumption is used to obtain a random monotonic polygon with vertex set S in $O(K)$ time, where $n < K < n^2$ is the number of edges in the visibility graph of the monotonic chain with vertex set S. We remark that the non-degeneracy assumption made in [29] is not a convenience but crucial for their algorithm. In this section, we consider the problem of removing this non-degeneracy assumption.

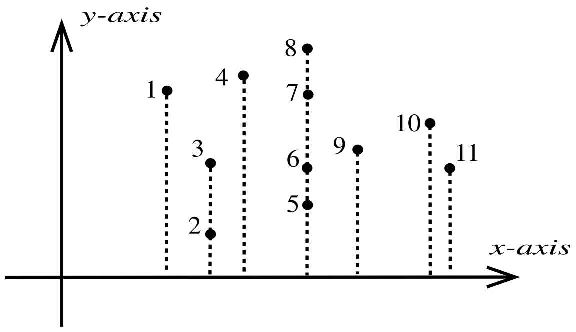

Figure 1: A non-regular projection of points on the x-axis.

A projection of S on a line L is said to be *regular* if each point in S projects to a distinct point on L, i.e., there are no vertical line degeneracies. In other words the projection contains n distinct points. Without loss of generality, when a line L is given, we will assume L to be the x-axis. If this is not the case, then in linear time, we may always rotate the configuration of points and line together so that L coincides with the x-axis. Figure 1 illustrates a set of points with a non-regular projection onto the x-axis.

2.1 The Decision Problem

We consider first the decision problem: given a set S, does it contain a degeneracy or does it give a regular projection on the x-axis. A very simple algorithm suffices since we simply need to check for duplicates in the x-coordinate values of the point set S. This can be done in $O(n \log n)$ time by sorting these values. The projection is regular if and only if there are no duplicates.

Furthermore, by a reduction from the element-uniqueness problem[1], a lower bound of $\Omega(n \log n)$ can be established for the regular projection decision problem: Given a set of n real numbers $A = \{a_1, \cdots, a_n\}$, the input S to the regular projection problem will be the set of planar points $\{(a_i, i) \mid 1 \leq i \leq n\}$. The elements of A are unique if and only if S has a regular projection on the x-axis. Thus we have the following.

Theorem 2.1 *Given a set S of n distinct points in the plane and a line L, whether S admits a regular projection onto L can be determined in $\Theta(n \log n)$ time.*

We should point out that the expected complexity of the algorithm can be reduced to $O(n)$ by allowing the use of floor functions as primitive operations and using a *bucket sorting* [14] algorithm.

2.2 The Computation Problem

Let us now turn to the computation problem: given a set S of n distinct points in the plane, find a rotation of S that removes the degeneracies, or equivalently, a line L such that S yields a regular projection on L. That a regular projection always exists follows from the fact that the only forbidden directions of projections are those determined by the lines through pairs of points of S. The forbidden directions are represented as follows: Let the circle C^2, representing the set of directions, be the unit circle in the plane, centered at the origin. For every pair of points in S, translate the line through them so that it intersects the origin. Intersect each such line with C^2 to yield a pair of forbidden points on C^2. We now have $n(n-1)/2$ not necessarily distinct forbidden points on C^2. Although there are $O(n^2)$ such forbidden directions, they have measure zero on the circle of directions C^2. Hence there is indeed no shortage of directions that allow regular projections. We may determine a direction for a regular projection by finding a point in the interior of any arc of C^2 that is bounded by two distinct adjacent forbidden points. Such a point on C^2 may be found easily in $O(n^2 \log n)$ time by using the brute-force approach of sorting the forbidden points.

Observe that using the $O(n \log n)$ time slope selection algorithm [11] to find the k-th and $(k+1)$-st slopes, which define an allowable interval of directions for obtaining a regular projection, will not necessarily lead to an improvement in run-time. This is because the slopes are not necessarily unique and hence the k-th and $(k+1)$-st slopes may have the same value. In fact, there may be several slopes, as many as $\Omega(n^2)$, with the same value.

[1]The *element uniqueness* problem is to determine, given an input set of real numbers, if any two of them are equal. This decision problem was shown to be $\Omega(n \log n)$ by Dobkin and Lipton [15].

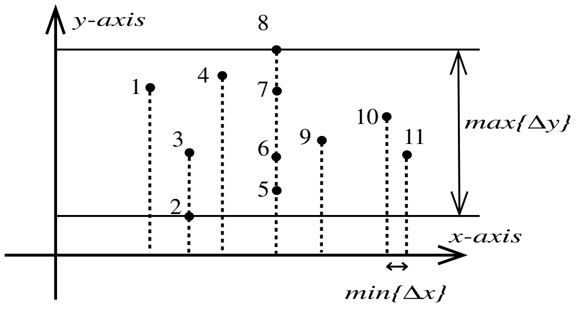

Figure 2: Computing a bound on the second largest distinct slope.

By combining the decision problem discussed above with a simple bounding technique we obtain an efficient algorithm for this problem. Given S, and our goal to determine a regular projection, first check if the vertical projection of S on the x-axis is regular. This takes $O(n \log n)$ time, as described in the previous section. If it is, we have the desired direction.

Hence assume that the projection of S onto the x-axis is *not* regular. This implies that the maximum slope, denoted by M_1, is equal to infinity. As mentioned above, looking for the next-largest non-duplicate slope, i.e, the maximum slope with the *constraint* that it not be infinite, could be costly. Let us denote this slope by M_2. The key realization for obtaining an efficient algorithm is that we do not need this constrained maximum slope. All we need is the value M of some slope such that $M_2 \leq M < M_1$, which may be found as follows (refer to Figure 2).

Let Δx and Δy denote, respectively, the absolute value of the difference in x and y coordinates of two distinct points in S. The slope of the line going through this pair of points in S is given by $\dfrac{\Delta y}{\Delta x}$. Let $\max_S\{\Delta y\}$ denote the maximum value that Δy can take over all pairs of points in S, i.e., the width of S in the y-direction. Similarly, let $\min_S\{\Delta x\}$ denote the minimum value that Δx can take over all pairs of points in S, i.e., the *smallest gap* in the x-coordinate values of the points in S. Of course, since the projection on the x-axis is not regular, we have that $\min_S\{\Delta x\} = 0$ (leading to an infinite maximum slope). To find a non-infinite upper bound on the second highest slope, first do the following: Sort the points in S by x-coordinate in order to remove all duplicates from the projected set of points on the x-axis. Let $S^* \subset S$ denote this reduced set of points with the property that no pair has the same x-coordinate. If there is just one point in S^*, then we have the case that all points are colinear on a line parallel to the y-axis. In such a situation, any line except the x-axis will give us a regular projection. In other words, the desired slope M can be any non-infinite value.

The desired direction (slope) for rotating S is then given by any finite value greater than $\frac{\max_S\{\Delta y\}}{\min_{S^*}\{\Delta x\}}$. Note that this bound is tight in the sense that the pair of points that determine $\min_{S^*}\Delta x$ could be the same pair that yields $\max_S\Delta y$, and hence S may contain a pair of points that realize the slope M, i.e., $M_2 \leq M$. Since $\min_{S^*}\{\Delta x\} > 0$ we have that $M < M_1 = \infty$. Computing $\max_S\{\Delta y\}$ takes linear time and computing $\min_{S^*}\Delta x$ takes linear time once S^* has been found, giving us the following result.

Theorem 2.2 *Given a set S of n distinct points in the plane, finding a regular projection onto the x-axis takes $O(n \log n)$ time.*

Once again, the expected complexity of the algorithm can be reduced to $O(n)$ by allowing the use of floor functions as primitive operations and using a *bucket sorting* algorithm [14] in the sorting step.

As an example of the application of our algorithm to remove degeneracies, let us return to the method of [29]. Our results imply that with a little extra work, their algorithm will run without having to invoke the assumption that no two points of S have the same x-coordinate. In particular, combining our results implies that we can solve the problem in $O(n \log n + K)$ time.

2.3 The Optimization Problem

We consider now the question of not merely removing the degeneracies (finding a regular projection), but removing them in the best way possible, i.e., we want to find the projection that is most robust or tolerant in a way that will be made precise below. One natural definition of tolerance is the idea, which comes from computer graphics, that the projected points are the result of viewing the points from some directional angle, perhaps by a viewer or a camera. Once the regular projection has been obtained it is desirable that the projection remain regular during subsequent perturbations of the position of the camera. In fact we would like to find, among all the regular projections, the one that has maximum tolerance in this sense, namely, the projection that allows the greatest angular deviation of the viewpoint without violating the regularity of the projection, i.e., without creating degeneracies. We call this the regular projection with *maximum projective tolerance.* The previous discussion implies that the solution to this problem is determined by the mid-point of the *largest gap* among consecutive forbidden points on C^2, the circle of directions. The largest gap can be found by sorting the $O(n^2)$ forbidden points.

A more general *weighted* version of this problem was first solved by Ramos in his thesis [31] on the tolerance of geometric structures [1]. He uses much more complicated techniques involving lower envelopes of trigonometric functions, but obtains the same time complexity as the above algorithm. It is interesting to note that such a simple algorithm exists for the unweighted case. The result is stated below.

Theorem 2.3 *Given a set S of n distinct points in the plane, a regular projection with the maximum projective tolerance can be computed in $O(n^2 \log n)$ time and $O(n^2)$ space.*

If the model of computation allows the use of the floor function as a primitive operation then there is a surprising and elegant linear time algorithm based on the "pigeon-hole" principle and due to Gonzalez [24, 30] for computing the largest gap of a set of numbers on the real line. This algorithm can be extended to work on the circle C^2, giving us an $O(n^2)$ time algorithm for computing a projection with the maximum projective tolerance.

3 No Two Points in Space on a Vertical Line

The remaining sections of this paper are concerned with removing non-degeneracy assumptions in 3-D (three-dimensional space), where there exists a greater variety of possible degeneracies than in the plane. In this section we consider the type of degeneracy that occurs when two points in space differ in only one of their coordinates, say the z-coordinate, i.e., when two (or more) points lie on a vertical line. This is one natural generalization of the planar degeneracy considered in the previous section.

Let S be a set of n distinct points in Euclidean space and let H be a plane. The definition of a regular projection in 3-D is similar to the one in the plane: a projection of S on H is said to be regular if each point in S projects to a distinct point on H, i.e., if no vertical line degeneracies occur when H is the xy-plane. Without loss of generality, when a plane H is given, we will assume H to be the xy-plane. It is convenient in 3-D to represent the directions of projection by points on the surface of the unit sphere centered at the origin, which we denote as C^3.

3.1 The Decision Problem

In the decision problem we wish to decide, given a set of points S in 3-D, if it contains any vertical-line degeneracies, i.e., if it gives a regular projection on the xy-plane. As in 2-D, a simple sorting algorithm gives the solution.

Let S_{xy} denote the set of points obtained by projecting S onto the xy-plane. First, sort the points in S_{xy} lexicographically by x and y coordinates. In other words, the points of S_{xy} are sorted by x-coordinate, and if two points have the same x-coordinate, they are ordered by their y-coordinates. Scan through the sorted list to determine whether two (or more) points in S_{xy} have the same x *and* y coordinates. If there are two such points, the projection on the xy-plane is not regular. If there are no such points, we conclude that there are no vertical line degeneracies in S. The run-time is dominated by the lexicographic sort step, which takes $O(n \log n)$ time.

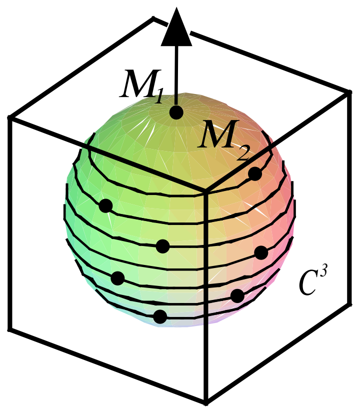

Figure 3: Computing a regular projection.

Once again, it is easy to establish a lower bound of $\Omega(n \log n)$ for this decision problem by showing a reduction from the element-uniqueness problem, for example. Thus we have the following.

Theorem 3.1 *Given a set S of n distinct points in space and a plane H, determining whether S admits a regular projection onto H is $\Theta(n \log n)$.*

As in the 2-D case we can obtain $O(n)$ expected time algorithms using bucket sorting if floor functions are allowed in the model of computation.

3.2 The Computation Problem

In this section, we want to compute, given a set S of n distinct points in space, a rotation of S that removes vertical-line degeneracies or equivalently, a plane H such that S yields a regular projection onto H. As in the two-dimensional case, the only forbidden directions of projections are given by lines in space going through pairs of points in S, because S does not admit a regular projection onto planes perpendicular to these lines. By translating every such line to the origin and intersecting it with the sphere of directions, we obtain $O(n^2)$ (not necessarily distinct) forbidden points on C^3.

We obtain an efficient algorithm for the computation problem in 3-D by applying a strategy similar to the one in 2-D. First check to see if the set S already projects regularly onto the xy-plane, which takes $O(n \log n)$ time. Assume now that this is not the case (i.e., the xy-plane is not a regular projection plane). This implies that there is at least one forbidden point at the "north-pole" of the sphere of directions C^3. Pairs of points in S that lead to a fixed slope in space correspond to points on "latitude" circles on C^3 (see Figure 3). The proper second highest slope corresponds to the smallest such circle containing the north pole (this is the "northernmost" latitude circle shown in Figure 3). To find a regular projection, it suffices to find a point other than the "north-pole" that lies in the interior of this smallest latitude circle.

Let S_{xy}, S_x and S_y denote the sets of planar points obtained by projecting S onto the xy-plane, x-axis and y-axis, respectively. Sort each of S_{xy}, S_x and S_y lexicographically. Scan through each sorted list to remove multiple occurrences of a point (keeping one copy). Call these new sorted lists S^*_{xy}, S^*_x and S^*_y, respectively. As before, let Δx, Δy and Δz denote the difference in x, y and z-coordinate values, respectively, of any two distinct points in 3-D.

The algorithm for the computation problem is as follows: Compute the width of S in the z-direction, i.e., $\max_S\{\Delta z\}$. Also compute $d = \min\{\min_{S^*_x}\{\Delta x\}, \min_{S^*_y}\{\Delta y\}\}$. The required slope M that lies between the largest slope (call this $M_1 = \infty$) and the next largest slope (M_2) is given by $M = \max_S\{\Delta z\}/d$. Observe that $d > 0$. We explain some details in the following paragraphs.

Consider a line going through two points in S. Its slope is given by $\Delta z/\sqrt{(\Delta x)^2 + (\Delta y)^2}$. Since the projection onto the xy-plane is not regular, it follows that there is a pair of points in S for which $(\Delta x)^2 + (\Delta y)^2$ is zero. The line through this pair of points has maximum slope M_1 $(= \infty)$, which leads to a forbidden point at the north-pole. Our objective is to find a non-infinite upper bound on the second-highest slope. In other words, we find a pair of points from S that maximizes Δz and a pair of points from S that gives the smallest possible non-zero value for $(\Delta x)^2 + (\Delta y)^2$. The former is given by a pair of points in S that define the width of S in the z-direction i.e., $\max_S\{\Delta z\}$, which can be computed in linear time. The latter is in fact given by the closest-pair in S^*_{xy}.

Computing the closest pair in S^*_{xy} in $O(n \log n)$ time requires some non-trivial method, such as Voronoi diagram construction. However, to keep the algorithm as *simple* as possible, we avoid the use of such a method. Observe that it is not necessary to actually find the closest pair - a positive value less than the closest-pair distance suffices. A simple argument shows that d is less than or equal to the closest-pair distance: If the closest pair of points in S^*_{xy} is either co-vertical or co-horizontal, then d will in fact be the closest-pair distance. If the closest-pair is neither co-vertical nor co-horizontal, then d will be *less* than the closest-pair distance. Therefore, $M = \max_S\{\Delta z\}/d$ gives us the required value lying between the largest (M_1) and second-largest (M_2) slope, i.e., $M_2 \leq M < M_1 = \infty$. Clearly M can be computed in $O(n \log n)$ time. S admits a regular projection onto a plane perpendicular to any line with finite slope greater than M.

Theorem 3.2 *Given a set S of n distinct points in space, finding a regular projection of S onto the xy-plane takes $O(n \log n)$ time.*

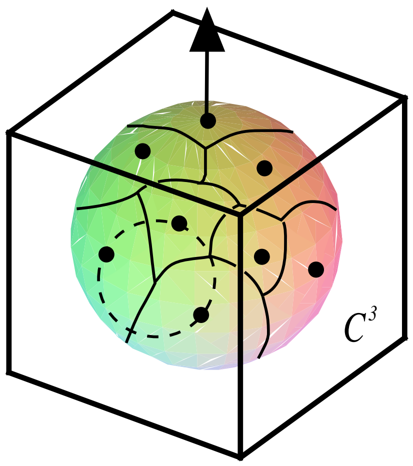

Figure 4: The largest empty circle.

3.3 The Optimization Problem

As in the two-dimensional case, it is desirable to remove vertical-line degeneracies in an optimal way, i.e., a direction of projection from which we can deviate the most without introducing degeneracies. As before, we call this a regular projection with the maximum projective tolerance.

The above discussion implies that the solution to this problem is determined by the center of the largest empty circle among the $O(n^2)$ forbidden (and not necessarily distinct) points on C^3, the sphere of directions (see Figure 4). We can find and discard all duplicate points on C^3 by performing a lexicographic sort of the forbidden points. Then the largest empty circle can be found by computing the Voronoi diagram of the remaining points on the sphere. Augenbaum and Peskin [4] give an $O(m^2)$ time insertion method for computing the Voronoi diagram of a set of m points on a sphere. There is a faster algorithm given by Brown [8], in which the Voronoi diagram is obtained in linear time from the convex hull of the set of points (treating them as a set of points in three space). The convex hull of a set of m points in three dimensions can be constructed in $O(m \log m)$ time (even in the presence of degeneracies such as four or more coplanar points; see [16]). We then have the following result.

Theorem 3.3 *Given a set S of n distinct points in space, a regular projection plane with the maximum projective tolerance can be computed in $O(n^2 \log n)$ time and $O(n^2)$ space.*

4 No Two Points in Space with the Same Coordinates

Many geometric algorithms dealing with point sets in 3-D assume that no two points have the same coordinate along one or more axes (see [13], for example). Note that in 2-D the assumptions that (1) no two points lie on a vertical line and (2) no two points have the same x-coordinate, are equivalent. In 3-D on the other hand, the two problems are different. A set of points in 3-D that admits a regular projection onto the xy-plane may still have points with the same x-coordinate. In this section we first consider the problem of rotating a set of points in 3-D so that no two points have the same x-coordinate. Then we show how to generalize this result so that no two points have the same coordinate for two or all three coordinate axes.

4.1 The Decision Problem

Given a set S of n distinct points and a line L in 3-D, the decision problem is to decide if S contains a degeneracy or if it admits a regular projection onto L. We may assume, without loss of generality, that L is the x-axis. By sorting the points of S by their x-coordinate value, we can determine if any two points of S have the same x-coordinate, i.e., if the projection onto the x-axis is regular or not. Once again, we reduce element-uniqueness to this problem to show an $\Omega(n \log n)$ lower bound.

Theorem 4.1 *Given a set S of n distinct points in space, whether S admits a regular projection onto the x-axis can be determined in $\Theta(n \log n)$ time.*

4.2 The Computation Problem

Given a set S of n distinct points in 3-D, the problem is to find a rotation of S that removes the degeneracies or, equivalently, find a line L such that S yields a regular projection onto L. This can be done as follows: first find a plane of regular projection for S. From Theorem 3.2 we know that this can be done in $O(n \log n)$ time. Let H be such a plane and let S_H be the planar set of points obtained by projecting S onto H. In the plane H, find a line of regular projection for S_H. From Theorem 2.2 we know that this can be done in $O(n \log n)$ time as well. Since there are no duplicates in S_H, this line is also a line of regular projection for S. Finally, we can translate and rotate the line so that it coincides with the x-axis.

In [13], it is actually assumed that the points have distinct coordinates in all their coordinate axes i.e. no two points have the same x, no two points have the same y and no two points have the same z-coordinate. We can extend our technique to handle this more general case as well, within the same time bound. First we obtain, using Theorem 3.2, a rotation of S such that the new rotated set S_r yields a regular projection onto the xy-plane. Next we seek a rotation of S_r about the z-axis so that the resulting rotated set S_r' has regular projections on *both* the x and y axes. We do this as follows: Regardless of whether the projection of S_r is or is not regular on the x-axis, we find (using Theorem 2.2) a bound δ_1 for the angle of rotation of S_r to achieve the desired result. Then we repeat this procedure to determine a second bound δ_2 for a positive angle of rotation to make the projection on the y-axis regular. Finally, we would like the rotated set S_r' to also have a regular projection on the zy-plane (a property we will need later). Therefore, using Theorem 3.2 again, we find a bound δ_3 for an angle of rotation to accomplish this task. Finally to obtain S_r' we rotate S_r about the z-axis by an angle $\delta = \min\{\delta_1, \delta_2, \delta_3\}$. We still need to ensure that there are no degeneracies along the z-axis. To do this we rotate S_r' about the x-axis. Note that since the previous rotation ensured S_r' has a regular projection on the zy-plane, we can do this. However, although such a rotation will not re-introduce degeneracies along the x-axis, it may do so along the y-axis. To prevent this, we proceed along lines similar to the first step: we find two bounds δ_4 and δ_5 for rotations which will ensure that the y and z axes, respectively, have no degeneracies and we rotate S_r' about the x-axis by the smaller of δ_4 and δ_5. Therefore we have the following result.

Theorem 4.2 *Given a set S of n distinct points in space, a rotation of S such that no two points have the same coordinate can be computed in $O(n \log n)$ time.*

4.3 The Optimization Problem

In this section, we give an algorithm for the optimization problem, i.e., for removing degeneracies in such a way that the resulting line L of regular projection is the most robust. The projection of S onto L will then be said to have the maximum projective tolerance. We make this precise in the remainder of this section.

Consider the direction orthogonal to a plane through a pair of points of S. In this axial direction the two points have the same coordinate. Therefore this is a forbidden direction for the final x-axis. This situation is true for every plane through the pair of points. Therefore, a pair of points in S yields a great circle on the sphere C^3 as the set of forbidden directions for the desired x-axis. Thus the entire set S yields an arrangement of $O(n^2)$ great circles on C^3. To find a regular projection we must find a point in the interior of any face of this arrangement.

To find a projection of maximum projective tolerance, we need to find a point on the sphere C^3 which is the center of the largest spherical disc contained in a face of the arrangement of great circles (see Figure 5). There are $O(n^2)$ great circles with a total of $O(n^4)$ intersections in the arrangement. Compute the entire arrangement and the largest spherical disc contained in each face, which takes time linear in the size of the face. It follows that a line L in space such that S yields the regular projection onto L with the maximum projective tolerance can be found in $O(n^4)$ time and space.

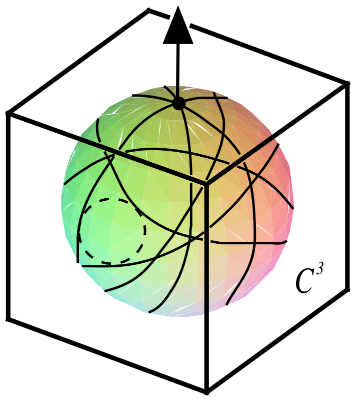

Figure 5: The largest circle contained in a face of the arrangement

Theorem 4.3 *Given a set S of n distinct points in space, a line L in space such that S yields a regular projection onto L with the maximum projective tolerance can be found in $O(n^4)$ time and space.*

It is possible, with a slight increase in run-time, to reduce the above space complexity to $O(n^2)$ as follows: Let G be the set of (forbidden) great circles on C^3 and let $\mathcal{A}(G)$ denote the arrangement of these great circles. For any great circle α on C^3, we have the following notation: $n(\alpha)$ denotes the line through the origin and normal to α. Let $d(p, \alpha)$ be the perpendicular distance from a point p on C^3 to the plane containing α and (by a slight abuse of notation) $d(p, l)$ the perpendicular distance from a point p to the line l through the origin. Let $N(\alpha)$ and $N'(\alpha)$ denote the two anti-podal points at which $n(\alpha)$ intersects C^3 and P be the set of all such points obtained from G i.e. $P = \{N(\alpha) | \alpha \in G\} \cup \{N'(\alpha) | \alpha \in G\}$. Finally, we use $\ell(p)$ to denote the line through the origin containing the point p on C^3.

The largest empty spherical disc contained in a face of $\mathcal{A}(G)$ is centered at a point on C^3 that maximizes the minimum distance from a point to the great circles in G. Note that for every point on C^3 that satisfies this condition, so does its anti-podal pair. Consider such a point p_m.

For a point p on C^3 and a great circle α, it is easy to see that $d(p, \alpha)$ increases as $d(p, n(\alpha))$ decreases. Furthermore, $d(p, n(\alpha)) = d(N(\alpha), \ell(p)) = d(N'(\alpha), \ell(p))$. In particular, the relation is $(d(p, \alpha))^2 + (d(p, n(\alpha)))^2 = (d(p, \alpha))^2 + (d(N(\alpha), \ell(p)))^2 = 1$. It follows from this observation that since p_m maximizes the minimum distance from a point to the great circles in G, therefore $\ell(p_m)$ minimizes the maximum distance from a line to the points in P.

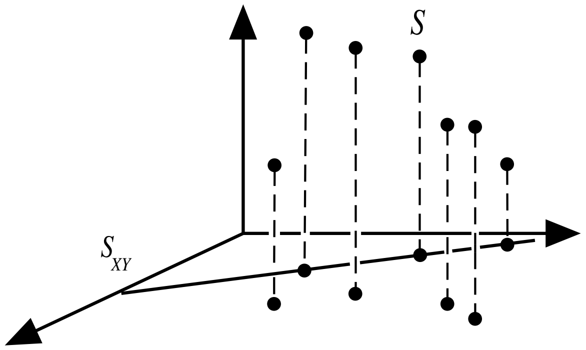

Figure 6: Non-colinear projections.

Therefore, computing the largest spherical disc contained in a face of the arrangement of G is equivalent to computing a line through the origin that minimizes the maximum distance of the points in P to that line. This is the problem of computing the *thinnest anchored cylinder* containing P. Follert [21] gives an $O(m\lambda_6(m)\log m)$ time and $O(m)$ space algorithm for this problem, where $\lambda_6(m)$ is the maximal length of any Davenport-Schinzel sequence of order six over an alphabet of size m and is known to be slightly superlinear [2]. In our case, $m = |P| = O(n^2)$.

5 No Three Points in Space on a Vertical Plane

Another non-degeneracy assumption that is commonly made for a set of points in three dimensions is that no three of them lie on a vertical plane. For example, Guibas et al. [3], give a linear-time algorithm for constructing the convex hull of a set of points in three dimensions with certain special properties and one of the assumptions in their algorithm is that no three of the input set of points lie on a vertical plane. Furthermore, some geometric algorithms use the projected points in intermediate steps, in which they use algorithms that make the non-degeneracy assumption that no three of the projected points are colinear. This assumption is equivalent to the assumption that the input set of points in three dimensions has no three on a vertical plane (see Figure 6). Therefore it is desirable to find directions of projection in which this degeneracy does not hold. In this section, we consider the problem of finding such a direction of projection and therefore, of removing this non-degeneracy assumption.

Let S be a set of distinct points in Euclidean space and let H be a plane. The projection of S onto H is said to be a *non-colinear projection* if no three of the projected points are colinear, i.e., there are no vertical plane degeneracies when H is the xy-plane. Note that each of the projected points must also be distinct, so that a non-colinear projection is also a regular projection.

5.1 The Decision Problem

Assume, without loss of generality, that H is the xy-plane. We would like to decide if the input set S of n points in space has vertical-plane degeneracies or if the projection onto the xy-plane is a non-colinear projection. To solve the decision problem, first determine if there are any vertical-line degeneracies using Theorem 3.1. If there are, then S has vertical-plane degeneracies. If there are not, first project S onto the xy-plane to obtain S_{xy} and find an orientation for the x-axis such that no two points in S_{xy} have the same x-coordinate, which takes $O(n \log n)$ time (see Theorem 2.2). Checking for colinearities can be done by using the dual map $(a, b) \mapsto y = ax + b$ of S_{xy}. The dual is an arrangement of n lines. In the arrangement, a vertex of degree six corresponds to three colinear points. Constructing the whole arrangement and checking for vertices of degree six or more can be done in $O(n^2)$ time and space (see [18]).

The problem of deciding whether three points from a set of points in space lie on a vertical plane admits the obvious reduction from the problem of deciding if there are three colinear points in a set of points in the plane (the 3-colinear problem). This reduction shows that the decision problem belongs to the class of 3SUM-hard problems [23]. Problems in the class are at least as hard as the problem of determining, given a set of integers, if there are three that sum up to zero. In [23], a linear time reduction from this base problem to the planar 3-colinear problem is shown. Hence we have the following.

Theorem 5.1 *Given a set S of n distinct points in space and a plane H, determining whether S admits a non-colinear projection onto H takes $O(n^2)$ time and space.*

5.2 The Computation Problem

In this section, we give an efficient algorithm to compute, for a given set S of n distinct points in space, a rotation of S that removes vertical-plane degeneracies; in other words, an algorithm to find a plane H such that S yields a non-colinear projection onto H.

Observe that if S has three (or more) colinear points, there are *no* non-colinear projections. It follows from this that the computation problem for finding a non-colinear projection is 3SUM-hard. This is because *any* algorithm for this computation problem must (a) either output a direction of non-colinear projection or (b) indicate that no such direction exists. In the former case, no three points in the input set are colinear and in the latter case, there must be at least three colinear points in the input set. Therefore, the 3-colinear problem for a set of points in space, which is 3SUM-hard, has a linear-time reduction to the problem of computing a direction of non-colinear projection. Three points in the input are colinear if and only if there is no direction of non-colinear projection.

We now give an algorithm for the computation problem. Assume that no two points in S have the same x coordinate. As shown in Section 4.2, we can find in $O(n \log n)$ time an orientation of the axes such that this assumption is true. It follows therefore that the north pole of the sphere of directions represents a regular projection plane.

First determine whether three or more points in S are colinear. If there are, indicate that no non-colinear projection exists and stop. Checking for colinearities among points in 3-D takes $O(n^2)$ time and space, as described below: Project S onto the xy-plane to obtain S_{xy}. If three points in S are colinear, their projected images in S_{xy} will also be colinear. However, the converse is not true i.e. if three (or more) points in S_{xy} are colinear, the corresponding three points in S are not necessarily colinear. Consequently, our goal is to check, for all colinear points in S_{xy}, whether the corresponding points in S are colinear or not. Given a set $S'_{xy} \subset S_{xy}$ of m colinear points, let $S' \subset S$ be the set of points that gives rise to S'_{xy}. Clearly all points in S' lie on a vertical plane. Therefore, we can check if three points in S' are colinear in $O(m^2)$ time, since it is a co-planar set of points (as described in Section 5.1, for instance). The algorithm then is the following: Construct the arrangement $\mathcal{A}(\mathcal{L})$ of the set of lines $\mathcal{L}$ given by the dual of S_{xy}. Note that the lines are distinct and no two are parallel. Let $\mathcal{V}$ be the set of vertices of $\mathcal{A}(\mathcal{L})$. All lines of $\mathcal{L}$ that go through a vertex v of $\mathcal{V}$ correspond to a set of $\deg(v)/2$ colinear points in S_{xy}, where $\deg(v)$ is the degree of vertex v. As pointed out earlier, the corresponding set of points in S can be tested for colinearities in $O((\deg(v))^2)$ time. Therefore, the total run-time of the algorithm is $\leq c \sum_{v \in \mathcal{V}} (\deg(v))^2$, where c is some positive constant. By using induction on the number of lines in the arrangement, it can be shown that this total is in fact $O(n^2)$.

In particular, we show that $\sum_{v \in \mathcal{V}} (\deg(v))^2 \leq 8n^2$. For the base case of $n = 2$, we have $\sum_{v \in \mathcal{V}} (\deg(v))^2 = 16$ and hence the claim is true. Assume now that the claim is true for $|\mathcal{L}| = n$. Given an arrangement $\mathcal{A}(\mathcal{L})$ of n lines, let a new line ℓ be added to this arrangement to give us an arrangement of $n+1$ lines. Of the set of vertices of $\mathcal{A}(\mathcal{L})$, let V_A be the subset of vertices that ℓ does *not* go through and let V_B be the subset that ℓ does go through. Let V_C be the new vertices created by the addition of ℓ. Note that the degree of every vertex in V_B increases by two in the new arrangement $\mathcal{A}(\mathcal{L} \cup \ell)$ and all the vertices in V_C have degree 4. The set $\mathcal{V}$ of vertices of $\mathcal{A}(\mathcal{L} \cup \ell)$ is simply the union of V_A, V_B and V_C. Therefore, $\sum_{v \in \mathcal{V}} (\deg(v))^2 = \sum_{v \in V_A} (\deg(v))^2 + \sum_{v \in V_B} (\deg(v)+2)^2 + \sum_{v \in V_C} (\deg(v))^2 = \sum_{v \in V_A} (\deg(v))^2 + \sum_{v \in V_B} (\deg(v))^2 + 4\sum_{v \in V_B} \deg(v) + \sum_{v \in V_B} 4 + \sum_{v \in V_C} 16$. By the inductive hypothesis, $\sum_{v \in V_A} (\deg(v))^2 + \sum_{v \in V_B} (\deg(v))^2 \leq 8n^2$. Observe also that

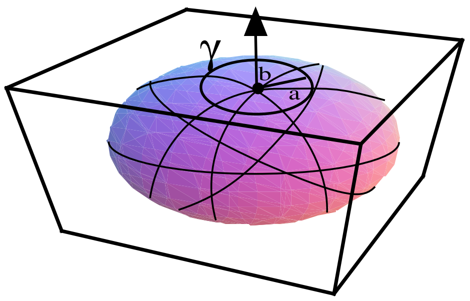

Figure 7: Computing a non-colinear direction of projection.

since the number of vertices that lie on ℓ can be at most n, we have $|V_B| + |V_C| \leq n$. Furthermore, we have $\sum_{v \in V_B} \deg(v) = 2(n - |V_C|)$, because each line in $\mathcal{L}$ that does not give rise to a new vertex ($\in V_C$) must contribute two towards this sum. Therefore, continuing the above equation, we have $\sum_{v \in \mathcal{V}} (\deg(v))^2 \leq 8n^2 + 8(n - |V_C|) + 4|V_B| + 16|V_C| \leq 8n^2 + 8n + 4|V_B| + 8|V_C| \leq$ (from the first observation stated above) $8n^2 + 16n \leq 8(n+1)^2$, which proves our claim. Therefore, checking for colinearities among n points in 3-D takes $O(n^2)$ time and space.

Suppose now that no three points in S are colinear. To see which directions are forbidden (for the plane normals), consider a triple of points in S. Each such triple defines a plane and every direction contained in that plane gives a forbidden direction (because the projection onto a plane perpendicular to that direction will have the said triple of points colinear). By intersecting those planes of forbidden directions with the sphere of directions we obtain a set of great circles. On C^3 we thus have $\binom{n}{3}$ great circles as the set of forbidden directions. To compute a non-colinear direction we must find a point that does not lie on any of the great circles. If S_{xy} is a non-colinear projection we are done. Otherwise, a certain number of great circles pass through the north pole of C^3. We denote by A the set of great circles containing the north pole (vertical great circles). For each great circle *not* in A we compute its distance to the north pole. The smallest such distance, which can be found in $O(n^3)$ time, determines a circle γ whose center is at the north pole (see Figure 7). Observe that γ has only vertical great circles intersecting it and is the largest such circle with center at the north pole.

Among all great circles in A we find the two distinct a, b that are adjacent when viewed from above. These can be found in $O(n^3)$ time. If c is a great circle through the north pole whose slope is between a and b, its intersection with γ leads to an open arc of non-colinear directions of projection. Note that if A has only one distinct great circle, i.e., $a = b$, then c can be *any* great circle through the north pole excepting a. Thus a non-colinear projection can be computed in $O(n^3)$ time and $O(n)$ space.

We show in the remainder of this section that this straightforward $O(n^3)$-time algorithm to find a direction of non-colinear projection can in fact be improved to an $O(n^2)$-time algorithm, but which requires $O(n^2)$ space. Since this computation problem is 3SUM-hard, it appears unlikely that this run-time can be improved. This improvement is obtained by making two observations that give us faster run-times for each of the two steps in the algorithm described above: (a) A pair of distinct great circles from the set A that are adjacent when viewed from above can, in fact, be found in $O(n^2)$ time, and (b) Instead of finding the largest circle centered at the north pole that does not contain any non-vertical great circles (the circle γ that we computed earlier), it is enough, in fact, to compute some circle *smaller* than γ, and this can indeed be done in $O(n^2)$ time.

Theorem 5.2 *Given a set S of n distinct points in space, deciding if a non-colinear projection exists and computing one, if it does, takes $O(n^2)$ time and space.*

Proof: As described above, deciding if there are three (or more) points of S that are colinear takes $O(n^2)$ time and space. If there are, indicate that no non-colinear projection exists and stop. If not, proceed with the remainder of the algorithm given below.

First we demonstrate (a) above. Consider the set A, i.e., all the great circles that go through the north pole. These great circles represent the set of all triples of points of S_{xy} that are colinear in the xy-plane. Consider the arrangement of lines in the dual of S_{xy}. Note that since we have ensured that no two points in S have the same x-coordinate, there will be no parallel lines in the dual of S_{xy}. Hence, three or more co-linear points in S_{xy} are represented in the dual by vertices that have degree six or more. In order to identify two adjacent great circles from A (when viewed from above), it is sufficient to identify the vertices of degree 6 (or more) in the arrangement that have the smallest and second smallest x-coordinate. Since the arrangement can be computed in $O(n^2)$ time, the first step of our algorithm has the same run-time. In other words, we have identified, in $O(n^2)$ time, two distinct great circles a and b that go through the north pole and that are adjacent.

Now let us proceed to (b), i.e., the question of computing a circle (call it δ) containing the north pole so that no non-vertical circles go through it. Let S be the input set of n points in 3-space. Consider any point of S and call it p. Of all the planes defined by triples of points in S, point p has $O(n^2)$ planes going through it. These planes define a set of $O(n^2)$ great circles on C^3: call this set G_p and let T_p be the *non-vertical* great circles from G_p. We find a circle δ_p centered at the north pole so that no great circle from T_p goes through it, and we do this for every p. The required circle δ will then simply be the smallest circle from the set $\{\delta_p \mid p \in S\}$.

We now describe a linear-time algorithm to find δ_p. For any $\alpha \in G_p$, let N_α be the point on C^3 that represents the normal to (the plane through) α. There are two points on C^3 that represent the normal to α; it suffices, for our purposes, to consider only those normal points that lie on or above the equator. If α is a *vertical* great circle, N_α will lie on the equator of C^3. In other words, the z-coordinate of N_α, $z(N_\alpha)$, will be zero. However, for every $\alpha \in T_p$, $z(N_\alpha)$ will be non-zero. To find a non-vertical plane that is closest to the north-pole, we need to find an N_α with the smallest z-coordinate greater than zero. This would then give us the *largest* circle centered at the north-pole so that no circle from T_p goes through it. In order to find a δ_p, however, it is sufficient to find *some* non-zero value that is smaller than $z(N_\alpha)$ for all $\alpha \in T_p$.

Consider now a great circle $\beta \in T_p$. β is given by a plane that goes through p and two other distinct points, say u and v. We denote by S' the set of points obtained from the input set S by identifying p with the origin and every other point with its central projection[2] on the unit sphere centered at the (new) origin. By a slight abuse of notation, we will continue to use the same symbol to refer to a point in S' as the corresponding point in S; hence, p ($\equiv (0,0,0)$), u and v now refer to points in S'. The great circle β then is given by a plane that goes through p, u and v.

Let $u \equiv (x_u, y_u, z_u)$ and $v \equiv (x_v, y_v, z_v)$. The cross-product of the vectors $\overrightarrow{pu}$ and $\overrightarrow{pv}$ gives us a vector N'_β normal to β. Assume, without loss of generality, that this vector lies above the equator. N'_β has the following coordinates:

$$N'_\beta \;=\; \overrightarrow{pu} \times \overrightarrow{pv} \;\equiv\; (y_u z_v - y_v z_u,\ x_v z_u - x_u y_v,\ x_u y_v - x_v y_u)$$

Observe that the coordinates of N'_β may not be equal to the coordinates of N_β. This is because N'_β is not necessarily a unit vector. However, the magnitude of N'_β is always *at most* one. This is an important observation, because it implies that $z(N_\beta)$ is always greater than or equal to $z(N'_\beta)$. Therefore, when we find a value less than $z(N'_\beta)$ for all such pairs of vectors, we have found a value less than $z(N_\alpha)$ for all $\alpha \in T_p$.

We show now that we can find, in linear time, a non-zero value less than $z(\overrightarrow{pu} \times \overrightarrow{pv})$, where $u, v \in S'$ are distinct and not colinear with p. Consequently, we obtain a δ_p in linear time. $z(N'_\beta) = x_u y_v - x_v y_u$, which is simply the area of the parallelogram P defined by the two vectors $\overrightarrow{pu}$ and $\overrightarrow{pv}$ when they are projected onto the xy-plane. Let l_u (l_v) be the length of the vector $\overrightarrow{h_u}$ ($\overrightarrow{h_v}$) obtained by projecting $\overrightarrow{pu}$ ($\overrightarrow{pv}$) onto the xy plane. The area of the parallelogram P is given by $l_u l_v \sin\theta$, where θ is the (smaller) angle between the two lines going through $\overrightarrow{h_u}$ and $\overrightarrow{h_v}$ respectively.

Let $l_{\min}$ be the length of the smallest vector among all such $\overrightarrow{h_u}$, $u \in S', u \neq p$. Let $\theta_{\min}$ be the smallest *non-zero* angle between any pair of lines given by the vectors $\overrightarrow{h_u}$. Clearly then $(l_{\min})^2 \ \sin\theta_{\min} \ \leq$ smallest non-zero area of all such parallelograms P,

[2]The *central projection* of a point (say s) on the unit sphere centered at the origin is the point of intersection between the directed half-line $\overrightarrow{s}$ and the unit sphere.

which is $\leq z(N_\alpha)$ for all $\alpha \in T_p$.

We can find $l_{\min}$ in linear time by projecting every point of S' onto the xy-plane, computing its distance to the origin and finding the smallest of these. It is also possible to find $\theta_{\min}$ in linear time from the arrangement $\mathcal{A}$ of lines given by the dual of S_{xy} (the set S projected onto the xy plane), which has already been constructed in step (a) of the algorithm. Let $\mathcal{L}$ be the line representing the dual of the point p' in S_{xy} obtained from p. By walking along $\mathcal{L}$ in $\mathcal{A}$, in linear time we obtain the complete sorted order of the lines going through the vectors $\vec{h_u}$. By scanning this sorted list, we find $\theta_{\min}$ in linear time.

■

5.3 The Optimization Problem

The problem of finding a direction of projection that allows the most deviation without introducing vertical-plane degeneracies, i.e., a projection with maximum projective tolerance, is reduced to the problem of finding the largest empty circle contained in a face of the arrangement of the forbidden great circles. Since there are $O(n^3)$ great circles in the arrangement, we have the following:

Theorem 5.3 *Given a set S of n distinct points in space, a direction such that S yields a non-colinear projection with the maximum projective tolerance can be computed in $O(n^6)$ time and space.*

As before, the technique outlined in Section 4.3 can be used to reduce the space complexity with a slight increase in run-time.

6 Removing Non-degeneracy Assumptions for Line Segments

In this section, we consider the problem of removing certain non-degeneracy assumptions for line segments. First we dispense with a straightforward non-degeneracy assumption, namely that no segment in the input set of n line segments is vertical. In 2-D and in 3-D, the decision problem is to check each segment until a vertical one, if it exists, is found. In 2-D, the slope of each segment defines a forbidden direction on the circle of directions C^2. In 3-D, we have n forbidden points on the sphere of directions C^3. The computation problem is also straightforward, where the goal is to find a non-vertical segment of largest slope. In 2-D, the slope of such a segment defines an arc of directions on C^2, any of which is a direction in which this non-degeneracy assumption is removed. Similarly in 3-D, a non-vertical segment of largest slope defines a spherical cap of directions centered at the north pole of C^3. The direction given by any point (other than the north pole) in the interior of this cap is an allowed direction of projection. Clearly the decision and computation problems take linear time in two and three dimensions. The optimization problems can also be solved as before: In 2-D, find the midpoint of the largest gap among the n forbidden points on C^2 and in 3-D, find the center of the largest empty circle among the n forbidden points on C^3. This can be done in $O(n \log n)$ time and $O(n)$ space.

In the remainder of this section, we consider the non-degeneracy assumption that no two line segments from a set of line segments in 3-D lie on a vertical plane. In 3-D computer graphics, the selection of a position of the eye, i.e., a direction of orthogonal projection is an important problem. For instance, in [26], Kamada and Kawai assert that the resulting image of the 3-D object, which is typically a "wire-frame" object made up of a collection of line segments, should be "easy to comprehend". This means that there should be as little loss of information as possible: information is lost, for example, when two line segments in space project to colinear planar line segments, or when line segments project to a single point. Therefore, when the line segments are projected onto the xy-plane, this condition is equivalent to the non-degeneracy requirement that no two line segments lie in a vertical plane. In [26], the authors call such a direction of projection (eye-position), in which degeneracies are removed, a "general eye-position".

In order to compute a "most general eye-position", i.e., a direction of projection in which such degeneracies are removed in the best way possible, Kamada and Kawai [26] give an algorithm that takes $O(n^6 \log n)$ worst-case time, where n is the size of the input set of line segments in space. Removing degeneracies in the "best way possible" is defined in a manner similar to the way in which we defined projections of maximum projective tolerance in earlier sections: this is the direction of projection that allows the most deviation without introducing any degeneracies of the type described above.

We give a more efficient algorithm for this problem by using the techniques described in this paper. Let S be the input set of line segments in 3-D (as in [26], assume no two line segments in space are colinear). Consider the plane, if it exists, containing a pair of line segments in S. Every line contained in this plane gives a forbidden direction of projection (an eye-position which is not a "general eye-position"). Therefore, this pair of line segments describes a great circle of forbidden directions on C^3. By doing this for every pair of line segments, we obtain an arrangement of $O(n^2)$ great circles on C^3.

We first briefly discuss the decision version of this problem, which can be solved in $O(n \log n)$ time as follows: If two or more segments in S are vertical, then clearly we have two segments on a vertical plane. If just one segment s is vertical, check in linear time if any other segment lies on a vertical plane with s. If not, consider the set of segments obtained by projecting the non-vertical segments onto the xy-plane and call this set S'. We want to check if two or more segments in S' are colinear. This can be done by lexicographically sorting the segments of S' by slope and when two segments s_1 and s_2 have the same slope m, ordering them by orthogonal distance from the line $y = mx$ through the origin (if s_1 lies in the half-plane $y < mx$, its distance from the line $y = mx$ is noted as negative and as non-negative otherwise).

To solve the computation version of the problem, we need to find a point that does not lie on any of the forbidden great circles on C^3. As described in Section 5.2, this can be done by finding two distinct consecutive great circles going through the north pole and the non-vertical great circle closest to the north pole (this defines, on C^3, a wedge of possible directions of projection such that no two segments lie on a vertical plane). From the discussion there, it follows that this can be done in $O(n^2)$ time and $O(n)$ space.

Finally we have the optimization problem: The "most-general eye position", a direction giving the projection of maximum projective tolerance, is given by the center of the largest spherical disc contained in a face of the arrangement of great circles. Therefore for the case of wire frame objects we have the following theorem:

Theorem 6.1 *Given a set S of n line segments in space, a direction such that S yields a projection with no two line segments colinear and with the maximum projective tolerance can be computed in $O(n^4)$ time and space.*

As in Section 4.3, let G be the set of great circles on C^3 and let P be the set of pairs of points on C^3 normal to each great circle in G. As stated there, the above problem is equivalent to computing the thinnest anchored cylinder of P and can be solved using Follert's algorithm [21]. This is equivalent to computing the two smallest anti-podal spherical caps containing P. There are applications, such as GIS and computer-aided architecture, in which it is desirable that the "eye-position" lies "above" every forbidden plane, as given by the great circles in G. In such cases, the problem reduces to finding the smallest enclosing spherical cap containing the points $P' \subset P$ that lie in the upper hemisphere. As explained below, this can be found by computing the smallest enclosing sphere of P' and intersecting it with C^3.

Let H be the unit hemisphere. Let P' be a set of n points on H. Let C be the minimum spanning circle of P', i.e., C corresponds to the boundary of the smallest spherical cap of H that contains P'. Let B be the smallest sphere that contains P'. Then

Lemma 6.1 $B \cap H = C$

Proof: Let S be the smallest sphere whose intersection with H is C. We will show that $S = B$. First we show that S contains P'. Since S is the smallest sphere whose intersection with H is C, it follows that C is a diametral or great circle of S. Therefore the diameter of S < diameter of H. Therefore P' is contained in S.

It remains to show that S is the smallest containing sphere of P'. Assume there exists a smaller containing sphere B'. Since the largest intersection that a sphere of the same size as S can make with H is C, it follows that the intersection of B' with H is smaller than C. But this contradicts the assumption that C is the minimum spanning circle.

■

From the above lemma, it follows that in order to compute the smallest enclosing spherical cap of P', we first compute the minimum enclosing sphere of the points in P' (considering them to be just a set of points in space) and then intersect this sphere with the sphere of directions C^3. The former can be done in $O(|P'|)$ time, as shown in [28], where Megiddo describes a linear-time algorithm for linear programming in three dimensions. The latter problem of computing the intersection of two spheres can be done in constant time, giving us an $O(n^2)$ time and space algorithm for this special case.

7 Summary of Results

We conclude by summarizing all the results obtained in the paper in the following two tables. The first table shows the results for point sets and the second one shows the results for line segments. In that table, we have also included the assumption that no face of a polyhedron in 3-D is vertical. This result follows immediately from the techniques described in the paper. Each face of the polyhedron describes a forbidden great circle of directions on the sphere of directions. Consequently, the decision and computation problems can be solved in linear time and the optimization problem can be solved using the technique described in Section 4.3.

Problem	Decision	Computation	Optimization
Point Sets			
2-D			
No two on a vertical line	$\Theta(n \log n)$	$O(n \log n)$	$O(n^2 \log n)$ time and $O(n^2)$ space
			$O(n^2)$ time and space with floor functions
3-D			
No two on a vertical line	$\Theta(n \log n)$	$O(n \log n)$	$O(n^2 \log n)$ time and $O(n^2)$ space
No two with the same x-coordinate	$\Theta(n \log n)$	$O(n \log n)$	$O(n^4)$ time and space
			$O(n^2 \lambda_6(n^2) \log n)$ time and $O(n^2)$ space
No two with the same x, y or z-coordinate	$\Theta(n \log n)$	$O(n \log n)$	OPEN
No three on a vertical plane	(3SUM-hard) $O(n^2)$ time and space	(3SUM-hard) $O(n^2)$ time and space	$O(n^6)$ time and space
		$O(n^3)$ time and $O(n)$ space	

Problem	Decision	Computation	Optimization
Line Segments			
2-D			
No vertical	$\Theta(n)$	$\Theta(n)$	$O(n \log n)$ time and $O(n)$ space
3-D			
No vertical	$\Theta(n)$	$\Theta(n)$	$O(n \log n)$ time and $O(n)$ space
No two on a vertical plane	$O(n \log n)$	$O(n^2)$ time and $O(n)$ space	$O(n^4)$ time and space
			$O(n^2 \lambda_6(n^2) \log n)$ time and $O(n^2)$ space
Faces			
No face of a polyhedron vertical	$\Theta(n)$	$\Theta(n)$	$O(n^2)$ time and space
			$O(n \lambda_6(n) \log n)$ time and $O(n)$ space

Finally, we mention some open problems. The optimization problem for the assumption that no two points in space have the same coordinates is open. We mentioned in the introduction that the techniques presented in this paper can also be used to find projections of points in 3-D that contain no four co-circular points, but in an impractical model of computation. Whether a simpler method can be designed to compute such "non-co-circular projections" is an interesting open problem.

Acknowledgments: The authors would like to thank Jean-Daniel Boissonnat, Hervé Brönnimann, Olivier Devillers, Fady Habra, Pedro Ramos, Monique Teillaud and Mariette Yvinec for useful discussions and references.

References

[1] M. Abellanas, J. Garcia, G. Hernandez, F. Hurtado, O. Serra, and J. Urrutia. Updating polygonizations. *Computer Graphics Forum*, 12(3):143–152, 1993.

[2] P. K. Agarwal, M. Sharir, and P. Shor. Sharp upper and lower bounds for the length of general davenport-schinzel sequences. *Journal of Combin. Theory, Ser. A*, 52:228–274, 1989.

[3] A. Aggarwal, L. J. Guibas, J. Saxe, and P. W. Shor. A Linear-Time Algorithm for Computing the Voronoi Diagram of a Convex Polygon. *Discrete and Computational Geometry*, 4:591–604, 1989.

[4] J. M. Augenbaum and C. S. Peskin. On the construction of the Voronoi mesh on a sphere. *Journal of Computational Physics*, 59:177–192, 1985.

[5] P. Bhattacharya and A. Rosenfeld. Polygons in three dimensions. *Journal of Visual Communication and Image Representation*, 5(2):139–147, June 1994.

[6] P. Bose, F. Gómez, P. Ramos, and G. Toussaint. Drawing nice projections of objects in space. In *Proceedings of Graph Drawing 95*, Passau, Germany, September 1995.

[7] P. Bose and G. Toussaint. Growing a tree from its branches. *Journal of Algorithms*, 19:86–103, 1995.

[8] K. Q. Brown. *Geometric transforms for fast geometric algorithms.* Ph.D. thesis, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, 1980. Report CMU-CS-80-101.

[9] C. Burnikel, K. Mehlhorn, and S. Schirra. On degeneracy in geometric computation. In *Proc. 5th ACM SIAM Symposium on Discrete Algorithms*, 1994.

[10] J. Canny, B. Donald, and E. K. Ressler. A rational rotation method for robust geometric computations. In *Proceedings of the 8th ACM Symposium on Computational Geometry*, 1992.

[11] R. Cole, J. S. Salowe, W. Steiger, and E. Szemeredi. An optimal time algorithm for slope selection. *SIAM Journal on Computing*, 18:792–810, 1989.

[12] G. B. Dantzig. *Linear Programming and Extensions.* Princeton University Press, Princeton, 1963.

[13] A. M. Day. The Implementation of an Algorithm to Find the Convex Hull of a Set of Three-Dimensional Points. *ACM Transactions on Graphics*, 9(1):105–132, 1990.

[14] L. Devroye and T. Klincsek. Average time behavior of distributive sorting algorithms. *Computing*, 26:1–7, 1981.

[15] D. Dobkin and R. Lipton. On the complexity of computations under varying sets of primitives. *Journal of Computers and System Sciences*, 18:86–91, 1979.

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

[17] H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 9(1):67–104, 1990.

[18] H. Edelsbrunner, J. O'Rourke, and R. Seidel. Constructing arrangements of lines and hyperplanes with applications. *SIAM J. Computing*, 15(2):341–363, 1986.

[19] I. Emiris and J. Canny. An efficient approach to removing geometric degeneracies. *Proc. 8th ACM Symposium on Computational Geometry*, pages 74–82, 1991.

[20] I. Z. Emiris and J. F. Canny. A general approach to removing degeneracies. *SIAM Journal of Computing*, 24(3):650–664, June 1995.

[21] F. Follert. Lageoptimierung nach dem maximin-kriterium. Master's thesis, Universität des Saarlandes, Saarbrücken, 1994.

[22] S. Fortune. *Directions in Geometric Computing*, chapter Progress in computational geometry, pages 82–127. Antony Rowe Ltd., Bumper's Farm, Chippenham, England, 1993.

[23] A. Gajentaan and M. H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational Geometry: Theory and Applications*, 5(3):165–185, 1995.

[24] T. Gonzales. Algorithms on sets and related problems. Technical report, Department of Computer Science, University of Oklahoma, 1975.

[25] M. Iri and K. Sugihara. Construction of the voronoi diagram for "one million" generators in single-presicion arithmetic. *Proceedings of the IEEE*, 80(9):1471–1484, September 1992.

[26] T. Kamada and S. Kawai. A Simple Method for Computing General Position in Displaying Three-Dimensional Objects. *Computer Vision, Graphics and Image Processing*, 41:43–56, 1988.

[27] C. Livingston. *Knot Theory*, volume 24 of *The Carus Mathematical Monographs.* The Mathematical Association of America, Washington, D.C., 1993.

[28] N. Megiddo. Linear-time Algorithms for Linear Programming in R^3 and Related Problems. *SIAM Journal on Computing*, 12(4):759–776, 1983.

[29] J. S. B. Mitchell, J. Snoeyink, G. Sundaram, and C. Zhu. Generating random x-monotone polygons with given vertices. In *Proceedings of the Sixth Canadian Conference on Computational Geometry*, pages 189–194, 1994.

[30] F. Preparata and M. I. Shamos. *Computational Geometry.* Springer-Verlag, 1985.

[31] P. Ramos. *Tolerancia de estructuras geometricas y combinatorias.* PhD thesis, Universidad Politecnica de Madrid, 1995.

[32] R. Seidel. The nature and meaning of perturbations in geometric computing. *Discrete and Computational Geometry*, 1996. to appear.

[33] C. K. Yap. A geometric consistency theorem for a symbolic perturbation scheme. *J. Computer and Systems Science*, 40:2–18, 1990.

[34] C. K. Yap. Symbolic treatment of geometric degeneracies. *Journal of Symbolic Computation*, 10:349–370, 1990.