The Complexity of Computing Nice Viewpoints of Objects in Space

Godfried T. Toussaint

ABSTRACT

A polyhedral object in 3-dimensional space is often well represented by a set of points and line segments that act as its features. By a *nice* viewpoint of an object we mean a projective view in which all (or most) of the features of the object, relevant for some task, are clearly visible. Such a view is often called a non-degenerate view or projection. In this paper we are concerned with computing non-degenerate *orthogonal* and *perspective* projections of sets of points and line segments (objects) in 3-dimensional space. We outline the areas in which such problems arize, discuss recent research on the computational complexity of these problems, illustrate the fundamental ideas used in the design of algorithms for computing non-degenerate projections, and provide pointers to the literature where the results can be found.

Keywords: Data visualization, nice viewpoints, knot theory, computer vision, graph drawing, computer graphics, degeneracies, general position, orthogonal projections, perspective projections, robustness of algorithms, computational geometry.

1. INTRODUCTION

In the scientific world we are frequently concerned with visualizing, describing and analyzing data consisting of three-dimensional (3-D) sets of points or line segments.[1] These points and/or line segments may be the data itself or they may have been extracted as fiducial features from a solid object such as a polyhedron. In this latter case the points may be some or all of the vertices of the polyhedron, and the line segments may be some or all of the edges of the polyhedron. However, we often have at our disposal only a 2-D medium, such as a computer-graphics screen or camera retina, on which to display a necessarily incomplete representation or picture of the object or data we are interested in. Therefore it is desirable to obtain 2-D representations of our data that make it easy for us to discover the structure in the real data in 3-D. This family of problems, the topic of this paper, falls in the areas of data visualization, knot theory, computer vision, computer graphics, graph drawing, and removal of degeneracies in computational geometry. These fields in turn belong to the much broader domain of scientific visualization,[2,3] the discipline concerned with helping us to better visualize information, objects and processes in their fullest generality.

One of the archetypal problems in graph-drawing[4,11] consists of asking, for a given graph, a *nice* drawing of it. A graph is not a rigid object in 3-D space but a more abstract topological structure which permits the shortening, lengthening and bending of its edges to achieve the desired goal. By contrast, in this paper we are more concerned with rigid metrical objects in 3-D space and we would like to obtain *nice* projections of these objects on some plane that will afford them. A class of problems more closely related to those considered here is concerned with obtaining nice projections of geometric graphs in space.[5] Such objects are rigid objects composed of points and line segments.

There are many ways to project an object onto a plane and it is therefore useful to distinguish between two fundamental and related methods: *perspective* projections and *orthogonal* (also orthographic or parallel) projections. A perspective projection represents a description of what the data look like from a particular point (location) in space. Orthogonal projections may be considered as perspective projections in the limit as the view point approaches a location infinitely far away from the data being viewed. Intuitively, we may think of our data as a set of points that block rays of light, embedded in 3-D space above the horizontal xy-plane, and the orthogonal projection of the data on the xy-plane as the shadow cast by the points when a light source shines from a point infinitely high along the positive z-axis. Obtaining nice orthogonal projections of the data then reduces to the problem of finding a suitable 3-D rotation for the points such that their shadow on the xy-plane contains the desired properties.

Keynote Address: Vision Geometry IX, Proc. SPIE International Symposium on Optical Science and Technology, 30 July to 4 August 2000, San Diego, California.

Painters in the fine arts are probably the earliest researchers of the methods for projecting the 3-D world we see onto the 2-D canvas. In 1435 Leone Battista Alberti[22] was the first to develop a logically coherent and pictorially adequate algorithm for obtaining perspective projections.[21]

There exists a variety of specific geometrical characteristics that can more accurately describe the vague notion of the *niceness* of a projection of an object. Some of these criteria are more desirable than others depending on the application in mind. For example, if the object is a set of points, one fundamental definition of nice is that all the points of the 3-D set should be visible in the projection. In other words, no point should be hidden behind another. In keeping with knot-theory notation,[20] we call such projections *regular*. Three natural computational problems immediately arise concerning regular (or any other *nice*) projections. The first is the *decision* problem: given a set S of n points in space and a plane H, determine if the projection of S onto H is regular. The second is the *computation* problem: given a set S of n points in space, find a plane on which the projection of S is regular if one exists. Finally, the third problem is the *optimization* problem: given a set S of n points in space, determine a projection that allows the greatest amount of perturbation of the viewpoint without ever causing a degeneracy. In other words, here we ask for the most robust (or stable) projection among all possible regular projections, given that they exist.

2. KNOT THEORY

2.1. Introduction

A knot is a non self-intersecting polygon in 3-D space.[19,20,39,40] A *knot-diagram* is a mapping of the knot from 3-D onto the plane. In order to manipulate the knot diagram from one configuration to another without changing the underlying knot, researchers need a diagram with certain key properties. The *regular* projection of a knot is a knot-diagram that contains the desired properties.

2.2. Regular Projections

The earliest work on non-degenerate orthogonal projections appears to be in the area of knot theory. Let S be a set of n disjoint line segments in E^3 specified by the cartesian coordinates of their end-points (vertices of S) and let H be a plane. Let SH be the orthogonal projection of S onto H. An orthogonal projection of S is said to be *regular* if no three points of S project to the same point on H and no vertex of S projects to the same point on H as any other point of S.[20] This definition implies that for disjoint line segments (1) no point of SH corresponds to more than one vertex of S, (2) no point of SH corresponds to a vertex of S and an interior point of an edge of S, and (3) no point of SH corresponds to more than two interior points of edges of S. Therefore the only crossing points (intersections) allowed in a regular projection are those points that belong to the interiors of precisely two edges of S. This condition is crucial for the successful visualization and manipulation of knots.[20] Knots, being non self-intersecting polygons in 3-D, are special cases of sets of line segments where not all the segments are disjoint. Note that a vertex where two edges are joined together in the case when the line segments form a 3-D polygon counts as (not two) but one vertex. In a regular projection of a knot a crossing point comes from precisely two edges of the polygon in 3-D, and the viewing direction then unambiguously determines which edge is in front of the other. This in turn allows the Reidemeister moves to be applied to change one knot diagram into another.

Regular projections of 3-D polygons were first studied by the knot theorist K. Reidemeister[37] in 1932 who showed that all 3-D polygons (knots) admit a regular projection and in fact almost all projections of polygons are regular. Reidemeister however was not concerned with computing regular projections. The computational aspects of regular projections of knots were first investigated by Bose et al.,[5] under the real RAM model of computation. See also the thesis by Francisco Gomez.[17] Given a polygonal object (geometric graph, wire-frame or skeleton) in three dimensional euclidean space (such as a simple polygon, knot, skeleton of a Voronoi diagram or solid model mesh), they consider the problem of computing several "nice" orthogonal projections of the object. One such projection, well known in the graph-drawing literature, is a projection with few crossings. They consider the most general polygonal object, i.e., a set of disjoint line segments. They show that given a set of n line segments in space, deciding whether it admits a crossing-free projection can be done in $O(n^2 \log n + k)$ time and $O(n^2)$ space, where k is the number of such intersections and $k = O(n^4)$. This implies for example that, given a knot, one can determine if there exists a plane on which its projection is a simple polygon, within the same complexity. Furthermore, if such a projection does not exist, a minimum-crossing projection can be found in $O(n^4)$ time and $O(n^2)$ space. They showed (independently of Reidemeister) that a set of line segments in space (which includes polygonal objects as special cases) always admits a regular projection, and that such a projection can be obtained in $O(n^3)$ time. A description of the set of all

directions which yield regular projections can be computed in $O(n^3 \log n + k \log n)$ time, where k is the number of intersections of a set of quadratic arcs on the direction sphere and $k = O(n^6)$. Finally, when the objects are polygons and trees in space, they consider monotonic projections i.e., projections such that every path from the root of the tree to every leaf is monotonic in some common direction on the projection plane. For example, given a polygonal chain P, they can determine in $O(n)$ time if P is monotonic on the projection plane, and in $O(n \log n)$ time they can find all the viewing directions with respect to which P is monotonic. In addition, in $O(n^2)$ time, they can determine all directions for which a given tree or a given simple polygon is monotonic.

3. COMPUTER VISION

3.1. Introduction

In the computer vision field there is both a theoretical[23] interest in nondegenerate projections and a pratical one.[6] The theoretical work resembles the work described in the previous section on knot theory in that it is assumed that the object consists of idealized points and line segments or polygons and polyhedra. A tool used for computing viewpoints from which the maximum number of faces of a solid polyhedron is visible, is the *aspect graph*.[14] Closer to the spirit of this paper, the Wirtinger projections of interest in computer vision are very closely related to the regular projections of knot theory.

3.2. Wirtinger Projections

That certain types of non-degenerate orthogonal projections of 3-D polygons always exist for some directions of projection was re-discovered by Bhattacharya and Rosenfeld[19] for a restricted class of regular projections known as *Wirtinger* projections. Recall that regular projections allow two consecutive non-collinear edges of a 3-D polygon to project to two collinear consecutive edges on H. Therefore some shape features of the polygon may be lost in regular projections. In knot theory this is not a problem but for visualization applications this may not be desirable. Those regular projections in which it is also required that no two consecutive edges of the 3-D polygon have collinear projections, are known as Wirtinger projections. Bhattacharya and Rosenfeld[19] did not address the algorithmic complexity of actually finding Wirtinger projections. Bose et al.,[5] study the complexity of computing a single Wirtinger projection as well as constructing a description of all such projections for the more general input consisting of disjoint line segments. These results include therefore results for 3-D chains, polygons, trees and geometric graphs in general. The description of all projections allows one to obtain Wirtinger projections that optimize additional properties. For example, one may be interested in obtaining the most tolerant projection in the sense that it maximizes the deviation of the view-point required to violate the Wirtinger property.

3.3. Object Reconstruction from Projections

A related problem in computer vision is that of reconstructing a 3-D object from a set of given projections. That the given projections be regular is helpful in solving this problem. Some results along these lines are available for polygons[19] and sets of points.[16]

3.4. Finite Resolution Models of View Degeneracy

View degeneracy is a central concern in robotics where a robot must navigate and recognize objects based on views of the scene at hand.[24,25] In the idealized world assumed in the previous sections, degenerate views are not much of a problem if a viewpoint is chosen at random, since almost all projections are not degenerate. On the other hand, real world digital cameras have a finite resolution and therefore view degeneracy can no longer be ignored.[8] Several researchers in computer vision observed experimentally that degenerate views occurred with surprising regularity.[26–29] Several strategies have also been investigated for controlling the camera motion in order to move out of a degenerate view.[30] Dickinson, Wilkes and Tsotsos[6] provide some definitions of degeneracies relevant to vision systems with finite resolution, develop a computational model for estimating the probability that a random view is degenerate, and provide some experimental results with a class of objects that indicate that a typical probability of a degenerate view is 0.2 and can be a high as 0.5, a surprisingly high value.

4. COMPUTER GRAPHICS

In computer graphics one is interested in vizualising objects well, and therefore *nice* views[31,32] and non-degenerate views[9] are major concerns. In vector graphics the objects are usually polygons in space and so the work there also resembles the work in knot theory and theoretical computer vision. For example, Kamada and Kawai[9] proposed a method to obtain nice projections by making sure that in the projection, parallel line segments on a plane in 3-D project as far away from each other as possible. Intuitively, the viewer should be as orthogonal as possible to every face of the 3-D object. Of course this is not possible and therefore they suggest minimizing (over all faces) the maximum angle deviation between a normal to the face and the line of sight from the viewer. They then propose an algorithm to solve this problem in $O(n^6 \log n)$ time, where n is the number of edges in the polyhedral object in 3-D. Gomez et al.,[7] reduce this complexity to $O(n^4)$ time. Furthermore, they show that if one is restricted to viewing an object from only a hemisphere, as is the case with a building on top of flat ground, then a further reduction in complexity is possible to $O(n^2)$ time.

5. GRAPH DRAWING

In many situations, especially objects which are graphs that contain many more edges than do skeletons of polyhedral objects, another requirement for effective visualization is simplicity. One frequently used measure of simplicity is the number of crossings of edges in the projection. It is desirable to obtain the projection that minimizes the number of crossings. If the minimum number of crossings is zero we call such projections crossing-free. In some applications we may have a 3-D directed tree as an object of interest. Such a tree may represent a system of veins in the human brain for example, where the direction of an edge represents the direction of blood flow in the corresponding vein segment. Here it is of interest to determine if there exists a projection such that all the directions of the edges of the tree are monotonically increasing in a specified direction on the projection plane. In general we call such projections monotonic projections. More specifically, a projection is monotonic if the projected image on the projection plane is monotonic. A planar polygonal chain is monotonic if there exists a direction such that every line orthogonal to this direction, that intersects the chain, yields one connected component as the intersection. A planar polygon is monotonic if it can be partitioned into two chains each of which is monotonic with respect to the same direction. A tree is monotonic if it contains a root and a direction such that all shortest paths from the root to the leaves are monotonic with respect to that direction. In Ref.[5] projections for objects which are sets of disjoint line segments, simple polygons, polygonal chains and trees, are investigated. Furthermore polynomial time algorithms are given for all the problems.

Of course the objects considered in Ref.[5] are geometric graphs and not general graphs. For general graphs the notions of minimum crossing drawings and monotonic drawings are classic visualization problems that have been well studied in the context of graph drawing.[41] The general question of given a graph, can one find an embedding in the plane that minimizes the number of crossing edges, is NP-complete.[42] In fact this problem is also NP-complete for a variety of special cases.[43] A lot of work has been done for drawing graphs in a monotonic way in the plane. These drawings are known in the graph-drawing literature as *upward planar drawings*. The general problem of determining for a given directed graph, whether it can be drawn in the plane such that every edge is monotonically increasing in the vertical direction and no two edges cross is also NP-complete, as is the problem of deciding if an undirected graph can be drawn in the plane such that every edge is a horizontal or vertical segment and no two edges cross.[44]

Traditionally graph drawing was done in the plane and hence the importance of minimizing the number of crossings. More recently however graph drawing research has explored drawing graphs in 3-D.[45] In 3-D other criteria for good drawings become important and particularly drawings that have good views in many projections. Eades, Houle and Webber[11] proposed three models of good viewpoints but the complexity of computing them is too high to be practical in many situations. This prompted Houle and Webber[12] to propose new criteria for best views and approximate but more efficient algorithms for computing them. See also the thesis of Richard Webber.[13]

6. REMOVING DEGENERACIES IN COMPUTATIONAL GEOMETRY

6.1. 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[36] has distinguished between intrinsic

or *problem-induced* and extrinsic or *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 collinear 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. 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 that such an input may contain.

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. In one method to remove 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. 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. For a review of perturbation schemes, see Emiris and Canny,[34] Yap,[36] and Emiris, Canny and Seidel.[35]

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 complicated.[33]

One may wonder if one can have one's cake and eat it too. Can we find the *exact* solution for the *original* problem with simple algorithms that handle degeneracies? The answer depends on the type of cake we want to eat. Gomez et. al.,[17,7] address an issue concerning the assumption of non-degeneracies which has received very 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, they 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. This approach not only handles algorithm-induced degeneracies via orthogonal projections but some problem-induced degeneracies as well with the aid of perspective projections.

Ref.[7] considers several non-degeneracy assumptions that are typically made in the literature, proposes efficient algorithms for performing the pre- and post-processing steps (suitable rotations) that remove these degeneracies, analyzes their complexity in the real RAM model of computation and, for some of these problems, gives lower bounds on their worst-case complexity. The assumptions considered in Ref.[7] include:

1. no two points in the plane have the same x and y-coordinates,
2. no two points in space lie on a vertical line (regular orthogonal projection),
3. no two points in space have the same x, y and z-coordinates,
4. no three points in space lie on a vertical plane,
5. no two line segments lie on a vertical plane.

Incorporating their 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 the increased complexity of computing the required rotations.

6.2. No Two Planar Points on a Vertical Line

In this subsection we illustrate the fundamental techniques used in the literature to remove algorithm-induced degeneracies. We use as a generic example a very simple version of the problem for a set of points in the plane. These techniques form the basis for solving more complicated problems in higher dimensions. We illustrate all three versions of the problem: decision, computation and optimization.

Many papers in computational geometry assume that no two points of a given planar set S have the same x-coordinate. This is especially true when a *line-sweep* technique is used to solve the problem. Thus this is a typical algorithm-induced non-degeneracy assumption. Recall that 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. Therefore we are concerned here with the simplest version of computing a *regular* projection of an object.

6.2.1. The Decision Problem

The decision problem is: given a set S, does it contain a degeneracy or does it give a regular projection on the x-axis. This question can be answered in $O(n^2)$ time by simply checking every pair of points to determine if they have the same x-coordinate or not. However, a much faster 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 in this sorted list.

Furthermore, by a reduction from the element-uniqueness problem we can derive a lower bound on the complexity of this problem. The element-uniqueness problem is to determine, given an input set of real numbers, if any two of them are equal. This decision problem is known to be $\Omega(n \log n)$. 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. Therefore we may conclude that 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.

6.2.2. 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 $O(n^2)$ forbidden points.

Observe that using an $O(n \log n)$ time slope selection algorithm 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.

By combining the decision problem discussed above with a simple bounding technique we obtain an efficient algorithm for solving the computation 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 earlier. If it is, we have the desired direction and we are done. 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.

Let Δx and Δy denote, respectively, 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 collinear 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 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. We can therefore conlcude that 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.

6.2.3. 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 (*stable* in the computer vision terminology) 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[38] on the tolerance of geometric structures. 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, as the one described above, exists for the unweighted case. In conclusion, 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.

It is worth noting that 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 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.

6.3. Perspective Projections and Intrinsic Degeneracies

6.3.1. Introduction

Intrinsic degeneracies cannot be removed by rotations of the input. If a set of points S in 3-D contains three collinear points then so does every orthogonal projection of S. This is where *perspective* projections come to the rescue. In this subsection we discuss the problem of removing intrinsic degeneracies for point sets by using *perspective* projections as the global transformation. Not all intrinsic degeneracies can be removed with perspective projections. Intrinsic degeneracies that can be removed via perspective projections are called *quasi-intrinsic degeneracies.*[18,15,10] For example, given a set of points on the xy-plane which has four or more cocircular points, we are interested in finding a center of projection and a plane such that the projection of that set of points does not contain four cocircular points. However, we are not interested in projecting on planes which are far away from the xy-plane since this could radically change the positions of the points. We must therefore look for planes that are close to the xy-plane.

Gomez et al.,[15] consider computing non-degenerate *perspective* projections of sets of points and line segments in 3-dimensional space. For sets of points they give algorithms for computing perspective projections such that (1) all points have distinct x-coordinates, (2) all points have both distinct x- and y-coordinates, (3) no three points in the projection are collinear, and (4) no four points in the projection are cocircular. For sets of line segments they present an algorithm for computing a perspective projection with no two segments parallel. All their algorithms have time and space complexities bounded by low degree polynomials.

6.3.2. No Two Projected Points have the Same x-Coordinate

Motivated in part by the problem of visualizing a point set in 3-D projected onto a plane, Gomez et al.,[7] solve the problem of finding the *orthogonal* projection of S that has the property that the projected points have distinct x, y and z coordinates. We now illustrate the *perspective* version of that problem for the special case of distinct x-coordinates only. We assume throughout that the plane of projection is the xy-plane.

Let $S = \{p_1, \cdots, p_n\}$ be a set of n distinct points in space and c be a center of projection. We denote by $S^* = \{p_1^*, \cdots, p_n^*\}$ the perspective projection of S from c. When necessary, we will underline the dependency of S^* on c by writing $S^*(c)$ or $p_i^*(c)$. If there are two points in S that determine a line parallel to the y-axis, then the perspective projection of S will contain points with the same x-coordinate. We can test for this in $O(n \log n)$ time and linear space by projecting S orthogonally onto the xz-plane and checking the resulting projected points for duplicates. Using a lexicographic sort (along the x and z coordinates) of the projected points, followed by a scan of the sorted list, we can determine whether there exist any such duplicates in $O(n \log n)$ time.

Given a set of n distinct points such that no two of them determine a line parallel to the y-axis, deciding whether a given projection on the xy-plane has distinct x-coordinates can be done in $\Theta(n \log n)$ time and $\Theta(n)$ space. It is sufficient to check for duplicate x-coordinates by sorting lexicographically the projected points on the xy-plane.

Before turning to the computation problem we dispense with the existence problem, namely: does a perspective projection of S with distinct x-coordinates always exist? A point is said to be a *forbidden point* if it produces a projection with non-distinct x-coordinates when projecting from it. A plane is called a *forbidden plane* if it contains a pair of points of S and intersects the xy-plane at a line parallel to the y-axis. Furthermore, every pair of points in S may yield a forbidden plane and there are no others. Since all forbidden points are contained in $O(n^2)$ planes, and these planes have measure zero in space, we conclude that there always exists a perspective projection whose x-coordinates are all distinct.

Let us now consider the computation problem: given a set S of n points in 3-D, find a center whose projection has distinct x-coordinates. In particular, in the following we show that given a set of n distinct points such that no two of them determine a line parallel to the y-axis, a projection with distinct x-coordinates can be computed in $O(n \log n)$ time and linear space.

If necessary, we first change the coordinate system so that S lies in the first octant (that is, all points have positive coordinates). Next we project S orthogonally onto the xy-plane, and taking that projection as a planar set, compute its perspective projection from the origin onto some line. If such a projection is not regular, then we will find a new center on the negative part of the x-axis which does produce a regular projection, and change the coordinate system again by translating S in the $+x$ direction so that this new center coincides with the origin. An $O(n \log n)$ time algorithm can be used to compute this new center.[10] We note that after these operations the z-axis cannot contain centers of projection that yield non-regular projections on the xy-plane because there is no plane containing the z-axis and more than one point of S.

The following algorithm finds a valid projection for S with a projection center on the positive z-axis. Actually, the algorithm not only computes a valid center, but also obtains an open line segment of valid centers.

Let z_0 be a positive number such that $z_0 > max\{z(p_i) \mid i = 1, \cdots, n\}$, where $z(p_i)$ is the z-coordinate of p_i, and $c(z_0) = (0, 0, z_0)$ is a point on the z-axis. Let us consider $S^*(z)$ the perspective projection of S from $c(z) = (0, 0, z), z \geq z_0$ onto the xy-plane. If $S^*(z_0)$ turns out to have all its x-coordinates distinct, then we are done. Therefore assume this is not the case. Then we know that in $S^*(z_0)$ there are at least two points with the same x-coordinate. When varying the parameter $z, z \geq z_0$ in a continuous way, the points of $S^*(z)$ also move in a continuous way. Let $c(z_1) = (0, 0, z_1)$ be the nearest forbidden point to $c(z_0)$. In the set $S^*(z_1)$, there must exist two points, $p_{i_j}^*(z_1), p_{i_{j+1}}^*(z_1)$, whose x-coordinates when sorted are equal. Since the order of the projected points along the x-axis changes only at a forbidden point on the z-axis when $c(z)$ varies, it follows that points $p_{i_j}^*(z_1)$ and $p_{i_{j+1}}^*(z_1)$ were

already consecutive at $c(z_0)$. Therefore, it suffices to consider only those planes passing through two consecutive points in the ordering by x-coordinate at z_0. For each one of those planes, we compute the intersection with the z-axis and select the nearest point $c(z_1)$ to $c(z_0)$. Any point of the open line segment determined by $c(z_1)$ and $c(z_0)$ is an allowed center of projection. The algorithm runs in $O(n \log n)$ time and uses $O(n)$ space.

Next we describe an alternate second approach for computing perspective projections with distinct x-coordinates also in $O(n \log n)$ time and linear space. Although this algorithm is more complicated up to a constant factor, unlike the previous algorithm, it yields an unbounded interval of valid projection centers located on the z-axis. This may prove useful for stability purposes in some applications in computer vision.[26]

This second algorithm is based on the observation that when we project a point set S from a projection center c on the z-axis onto the xy-plane, two points p_i and p_j of S project to points with the same x-coordinate if and only if, in the orthogonal projection S' of S onto the xz-plane, we have that the projected points p'_i and p'_j are collinear with the projection center c. So, to compute a perspective projection of the point set S with distinct x-coordinates, it suffices to determine a point in the z-axis that is non collinear with any two points of S'. This is only possible when the points of S' are distinct.

The algorithm uses the results from Gomez et al.,[7] for computing regular orthogonal projections of planar points onto a line, and of 3-dimensional points onto a plane.

As with the first algorithm we start by changing the coordinate system, if necessary, so that S lies strictly in the first octant of the coordinate space. Then we rotate S so that the *orthogonal* projection S' of S onto the xz-plane is a regular projection, i.e., the points of S' are distinct. This can be done in $O(n \log n)$ time.[7] Let S'' be the projection of S' onto the x-axis. Note that S'' may have duplicates. Now perform a lexicographic sorting of S'' to first delete duplicates and then compute the smallest (non-zero) gap between an adjacent pair in the points of S'' that are not discarded. Call this gap G_{x-min}. This step is also $O(n \log n)$. Next compute the difference in z-coordinates between the highest and lowest (maximum and minimum z-coordinates) points in S'. Call this difference H_{z-max}. Let S_{max} equal the ratio of H_{z-max} over G_{x-min}. Note that S_{max} is an upper bound on the maximum slope (nearest to vertical) determined by any pair of points in S'. This follows from the fact that S' is a regular orthogonal projection of S.

Finally, let p_0 denote the point on the xz-plane whose x and z coordinates are equal to the maximum x and z coordinates, respectively, of the points in S'. We construct a line with slope equal to the negative of S_{max} such that it passes through p_0 and we compute the intersection point c_0 that this line makes with the z-axis. By this construction any point on the z-axis above c_0 cannot be collinear with two points of S'. Therefore the perspective projection of S from any such center above c_0 will not have two points with the same x-coordinate.

7. CONCLUDING REMARKS

Almost all the complexity results discussed in this paper are restricted to inputs of idealized points and line segments viewed under the real RAM model of computation. This may be appropriate for some areas such as knot theory and vector graphics. For other areas, such as computer vision, such an assumption implies infinite resolution images, an unrealistic assumption. One approach to solving this more practical problem would be to give some thickness to the objects, i.e., consider the points as little balls and the edges of the polygons as thin cylinders, and then to re-design the algorithms accordingly. This may turn out to be rather expensive. In practice it may be much more efficient to perform a half-dozen *random* rotations to obtain a nice projection. After all, for many problems in the idealized infinite precision model, a single random rotation yields a nice projection with probability one. Computing optimal projections on the other hand is another matter. Here approximate algorithms may yield efficient solutions that are also near-optimal, but these are open problems for future investigation.

ACKNOWLEDGMENTS

The author would like to thank Jit Bose, Peter Eades, Paco Gomez, Michael Houle, Ferran Hurtado, Suneeta Ramaswami, Pedro Ramos and Toni Sellarès for many useful discussions on these topics.

REFERENCES

1. A. Getis and B. Boots, *Models of Spatial Processes*, Cambridge University Press, 1978.
2. P. Keller, and M. M. Keller, *Visual Cues: Practical Data Visualization*, IEEE Computer Society Press, 1993.
3. R. S. Gallagher, Ed., *Computer Visualization: Graphic Techniques for Engineering and Scientific Analysis*, IEEE Computer Society Press, 1995.
4. G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*, Prentice Hall, 1999.
5. P. Bose, M. F. Gómez, P. Ramos and G. Toussaint, "Drawing nice projections of objects in space", *Proc. Graph Drawing'95*, pp. 52–63, Germany, 1995. *Journal of Visual Communication and Image Representation* **10**, pp. 155–172, 1999.
6. S. J. Dickinson, D. Wilkes, and J. K. Tsotsos, "A computational model of view degeneracy," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **21**, pp. 673–689, 1999.
7. F. Gómez, S. Ramaswami, and G. Toussaint, "On removing non-degeneracy assumptions in computational geometry", *Proc. of the 3rd Italian Conference on Algorithms and Complexity*, 1997.
8. J. Kender and D. Freudenstein, "What is a degenerate view?" *Proc. 10th International Joint Conference on Artificial Intelligence*, pp. 801–804, 1987.
9. T. Kamada, and S. Kawai, "A simple method for computing general position in displaying three-dimensional objects," *Computer Vision, Graphics and Image Processing* **41**, pp. 43–56, 1988.
10. J. A. Sellarès, "Structural and algorithmic aspects of geometric projections," *Ph.D. Dissertation*, Universitat Politecnica de Catalunya, Barcelona, Spain, 2000.
11. P. Eades, M. E. Houle, and R. Webber, "Finding the best viewpoint for three-dimensional graph drawings," *Proc. Graph Drawing'97*, Springer-Verlag, pp. 87–98, 1997.
12. M. E. Houle, and R. Webber, "Approximation algorithms for finding best viewpoints," *Proc. Graph Drawing'98*, McGill University, Montreal, Canada, Springer-Verlag, pp. 210–223, 1997.
13. P. R. Webber, "Finding the best viewpoints for three-dimensional graph drawings," *Ph.D. Thesis*, University of Newcastle, Australia, 1999.
14. H. Plantinga and C. R. Dyer, "Visibility, occlusion and the aspect graph," *International Journal of Computer Vision*, **5**, pp. 137–160, 1990.
15. F. Gómez, F. Hurtado, A. A. Sellarès, and G. T. Toussaint, "Perspective projections and removal of degeneracies," *Proc. of the Tenth Canadian Conference on Computational Geometry*, pp. 100–101, 1998.
16. F. Gómez, F. Hurtado, and G. T. Toussaint, "Nice projections and reconstruction of objects," *Proc. Second International Conference on Mathematics and Design*, San Sebastián, Spain, pp. 441–450, June 1-4, 1998.
17. F. Gómez, "Aspectos geometricos y computacionales de las proyecciones," *Ph.D. Thesis*, Universidad Politecnica de Madrid, Spain, 1996.
18. F. Hurtado, and A. A. Sellarès, "Proyecciones perspectivas regulares: Correspondencia por proyeccion perspectiva entre configuraciones planas de puntos," *Actas VII Encuentros de Geometria Computacional*, Madrid, Spain, pp. 57–70, July 7-9, 1997.
19. P. Bhattacharya, and A. Rosenfeld, "Polygons in three dimensions," *Journal of Visual Communication and Image Representation*, **5**, pp. 139–147, 1994.
20. C. Livingston, *Knot Theory*, Mathematical Association of America, 1993.
21. W. M. Ivins, Jr., *On the Rationalization of Sight*, Da Capo Press, Inc., 1973.
22. L. B. Alberti, *Della Pittura Libri Tre*, 1435.
23. J. B. Burns, R. S. Weiss, and E. M. Riseman, "View variation of point-set and line-segment features," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **15**, pp. 51–68, January 1993.
24. S. J. Dickinson, A. Pentland, and A. Rosenfeld, "3D shape recovery using distributed aspect matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **14**, pp. 174–198, February 1992.
25. S. J. Dickinson, A. Pentland, and A. Rosenfeld, "From volumes to views: An approach to 3D object recognition," *CVGIP: Image Understanding* **55**, pp. 130–154, 1992.
26. D. Weinshall and M. Werman, "On view likelihood and stability," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **19**, pp. 97–108, February 1997.

27. D. Wilkes, S. J. Dickinson, and J. K. Tsotsos, "A quantitative analysis of view degeneracy and its application to active focal length control," *Proc. International Conference on Computer Vision* Cambridge, Massachusetts, June 1995.
28. D. Wilkes, S. J. Dickinson, and J. K. Tsotsos, "Quantitative modeling of view degeneracy," *Proc. Eighth Scandinavian Conference on Image Analysis* University of Tromso, Norway, May 1993.
29. I. Shimshoni and J. Ponse, "Finite resolution aspect graphs of polyhedral objects," *Proc. IEEE Workshop on Qualitative Vision* New York, pp. 140–150, June 1993.
30. S. J. Dickinson, H. Christensen, J. K. Tsotsos, and G. Oloffson, "Active object recognition integrating attention and viewpoint control," *Computer Vision and Image Understanding*, **67**, pp. 239–260, September 1997.
31. C. Colin, "Automatic computation of a scene's good views," *Proc. MICAD'90* 1990.
32. S. E. Chen and L. Williams, "View interpolation for image synthesis," *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH'93)*, pp. 279–288, 1993.
33. C. Burnikel, K. Mehlhorn, and S. Schirra, "On degeneracy in geometric computation," *Proc. 5th ACM SIAM Symposium on Discrete Algorithms*, pp. 16–23, 1994.
34. I. Z. Emiris and J. F. Canny, "A general approach to removing degeneracies," *SIAM Journal of Computing*, **24**, pp. 650–664, 1995.
35. I. Z. Emiris, J. F. Canny, and R. Seidel, "Efficient perturbation for handling geometric degeneracies," *Algorithmica*, **19**, pp. 219–242, 1997.
36. C. K. Yap, "Symbolic treatment of geometric degeneracies," *Journal of Symbolic Computation*, **10**, pp. 349–370, 1990.
37. R. Reidemeister, *Knottentheorie*, Ergebnisse der Mathematic, Vol. 1, Springer-Verlag, Berlin, 1932; L. F. Boron, C. O. Christenson and B. A. Smith (English translation) Knot Theory, BSC Associates, Moscow, Idaho, USA, 1983.
38. P. Ramos, "Tolerancia de estructuras geometricas y combinatorias," *Ph.D. Thesis*, Universidad Politecnica de Madrid, 1995.
39. K. C. Millet, "Knotting of regular polygons in 3-space," *Journal of Knot Theory and its Ramifications*, **3**, pp. 263–278, 1994.
40. G. T. Toussaint, "The Erdős-Nagy theorem and its ramifications," *Proc. 11th Canadian Conference on Computational Geometry*, Vancouver, Canada, pp. 9–12, August 16-18, 1999.
41. G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis, "Algorithms for drawing graphs: an annotated bibliography," *Computational Geometry: Theory and Applications*, **4**, pp. 235–282, 1994.
42. M. R. Garey, and D. S. Johnson, "Crossing number is NP-complete," *SIAM J. Alg. Discrete Methods*, **4**, pp. 312–316, 1983.
43. F. Shahrokhi, L. Szekely, and I. Vrt'o, "Crossing number of graphs, lower bound techniques and algorithms: A survey," *Lecture Notes in Computer Science*, **894**, Princeton, New Jersey, pp. 131–142, 1994.
44. A. Garg, and R. Tamassia, "On the computational complexity of upward and rectilinear planarity testing," eds., R. Tamassia and I. G. Tollis, *Proc. Graph Drawing'94, Lecture Notes in Computer Science*, **894**, Springer-Verlag, 286–297, 1995.
45. R. F. Cohen, P. D. Eades, T. Lin, and F. Ruskey, "Three-dimensional graph drawing," *Algorithmica*, **17**, pp. 199–208, 1997.