M.Sc. Engineering Thesis

Computing Nice Projections of 2D and 3D Scene

by
S. M. Shahriar Nirjon
Student No. 100605056P

Submitted to

Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science and Engineering

Department of Computer Science and Engineering (CSE)
Bangladesh University of Engineering and Technology (BUET)
Dhaka–1000, Bangladesh

The thesis titled "Computing Nice Projections of 2D and 3D Scene", submitted by S. M. Shahriar Nirjon, Student No. 100605056P, Session October 2006, to the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, has been accepted as satisfactory in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Engineering and approved as to its style and contents. Examination was held on July 23, 2008.

1. ____________________

Dr. Masud Hasan — Chairman
Assistant Professor — (Supervisor)
Department of CSE, BUET, Dhaka

2. ____________________

Dr. Md. Saidur Rahman — Member
Professor and Head — (Ex-officio)
Department of CSE, BUET, Dhaka

3. ____________________

Dr. Md. Mostofa Akbar — Member
Associate Professor
Department of CSE, BUET, Dhaka

4. ____________________

Dr. Mahmuda Naznin — Member
Assistant Professor
Department of CSE, BUET, Dhaka

5. ____________________

Dr. Md. Haider Ali — Member
Professor — (External)
Department of CSE, University of Dhaka.

Candidate's Declaration

This is to certify that the work entitled "Computing Nice Projections of 2D and 3D Scene" is the outcome of the investigation carried out by me under the supervision of Dr. Md. Masud Hasan in the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka 1000, Bangladesh. It is also declared that this thesis or any part of it has not been submitted elsewhere for the award of any degree or diploma.

এস.এম. শাহরিয়ার নির্জন

S. M. Shahriar Nirjon

Acknowledgment

All praises due to Allah, Lord of the Worlds, who granted me the ability to finish this thesis.

I would like to express my heartfelt gratitude to Dr. Masud Hasan, my thesis supervisor, for his continuous patience, guidance and encouragement. It was really a honour for me to have the opportunity to work with such a great researcher like him. Most of the works of this thesis were the results of joint work with him.

I would also like to thank Dr. Md. Kaykobad for his time to time advice and valuable comments. I am also thankful to the members of board of examiners– Dr. Md. Saidur Rahman, Dr. Md. Mostofa Akbar, Dr. Mahmuda Naznin and Dr. Md. Haider Ali for their valuable suggestions, advice and corrections.

Abstract

Computing nice projections of objects in 2D and 3D is a well studied problem in computational geometry. By nice projections we mean optimal projections having some special geometric property. Aside from theoretical interest, its application reaches in the domain of computer graphics, computer vision, object recognition, 3D graph drawing, visualization, robotics, knot theory etc. Considerable amount of research work has been done based on different criteria of niceness. For 3D objects some common criteria of niceness include maximizing (minimizing) the area of projection, minimizing the number of crossing in the projection of 3D lines, minimum overlapping among line segments and vertices, monotonicity of polygonal chains and generating silhouettes which meet some predefined criteria.

However, computing orthogonal projections of a set of line segments in 2D and 3D with the following optimality criteria have not been considered so far: (i) sum of the projected length of the line segments in 2D is the minimum and maximum, (ii) sum of the projected area of the triangles in 3D is the minimum and maximum, (iii) sum of the projected length of the line segments in 3D is maximum and minimum. (iv) maximizing (minimizing) the minimum (maximum) ratio between actual length and projected length of line segments in 2D. This thesis addresses these four problems and gives separate algorithms for each.

The underlying concept for Problem(i) and (ii) are similar. Here the idea of McKenna-Seidls algorithm is used by extending the concept of view from convex polyhedra to 2D and 3D scene. We have developed an $O(n \log n)$ algorithm for finding an optimal direction in Problem(i) and an $O(n^2)$ algorithm for that in Problem(ii). For problem (iii) we give several approximation algorithms. Experimental result shows that our algorithm is within constant factor of the optimum solution. In addition to our main objective on this problem, the above algorithm can be used in a novel application, which is to find the maximum (minimum) perimeter of a convex polyhedron in an orthogonal projection and for which no solution is known. The running time of this algorithm is $O(n^2)$. For Problem (iv), we developed an $O(n \log n)$ algorithm to find an optimal solution.

Contents

Acknowledgment 4
Abstract 5

1 Introduction **11**
1.1 The problems 14
1.2 Outline of the thesis 15

2 Preliminaries **17**
2.1 Orthogonal and perspective projection 17
2.2 Convex Polygon, polyheron and polytope 17
2.3 View and view cone 18
2.4 Spherical coordinates 20
2.5 Geodesic distance 21
2.6 Visibility ratio 21

3 Optimal projection of 2D and 3D scene **23**
3.1 McKenna and Seidel's approach 23
3.1.1 The problem 23
3.1.2 The solution 24
3.2 Lines in 2D 28
3.2.1 The problem 28
3.2.2 Concept of view 29
3.2.3 Forming the view cone 30
3.2.4 Finding the optimal point 31
3.3 Triangles in 3D 35
3.3.1 The problem 35
3.3.2 Concept of view and view cone 35

3.3.3 Finding the Optimal point . . . 37

4 Lines in 3D **43**
4.1 The problem . . . 43
4.2 The solution . . . 43
4.2.1 Concept of view . . . 44
4.2.2 The heuristic for maximization case . . . 45
4.2.3 The heuristic for minimization case . . . 49
4.2.4 Experimental results for maximization problem . . . 51
4.2.5 Experimental results for minimization problem . . . 57
4.3 A novel application . . . 59

5 Optimal visibility ratio for line segments in 2D **61**
5.1 Maximizing the minimum visibility ratio . . . 61
5.1.1 The problem . . . 61
5.1.2 Mapping the problem to unit circle . . . 62
5.1.3 Finding the optimal direction . . . 62
5.2 Minimizing the maximum visibility ratio . . . 64
5.2.1 The problem . . . 64
5.2.2 Mapping the problem to unit circle . . . 64
5.2.3 Finding the optimal direction . . . 65

6 Conclusion **68**
6.1 Future works . . . 69

References **70**

Index **72**

Glossary **75**

List of Figures

1.1 Minimum and maximum area projections. 13
1.2 Minimum crossing projections of 3D line segments. 14
1.3 Silhouette of projection with certain criteria. 15
1.4 Projection where the minimum visibility ratio is maximum. . . 16

2.1 Different types of projections. 18
2.2 Convex polygon and polyhedron. 18
2.3 Formation in view cone by line segments in 2D. 19
2.4 Formation in view cone by planes in 3D. 20
2.5 Spherical coordinates of a point. 21
2.6 Geodesic distance between two points. 22
2.7 Visibility ratio of line segments in 2D. 22

3.1 Minimum and maximum area projections. 24
3.2 Formation of visibility set. 25
3.3 Halfplanes dividing 3D space into conical regions. 26
3.4 Plane that is a part of convex polyhedron. 26
3.5 Formation of convex polyhedron. 28
3.6 The maximum and minimum sum projection of a set of line segments. 29
3.7 Formation of conical regions. 30
3.8 The line where the optimal point lies. 31
3.9 Convex polyhedron with inscribed and circumscribed circles. . 33
3.10 Triangles in 3D. 36
3.11 Every triangle has two normals. 36
3.12 Formation of conical regions. 37
3.13 The plane where the optimal point lies. 38

3.14 Convex polyhedron with circumscribed and inscribed spheres. 40

4.1 Projection of line segments in 3D. 44
4.2 Plot of the function to optimize. 45
4.3 Planes parallel to line segments. 46
4.4 Heuristics for trivial cases. 47
4.5 Heuristic for general case. 48
4.6 Bounding the direction. 48
4.7 Accuracy of our algorithm within a bounded cone. 52
4.8 Angular deviation of the heuristic based direction. 53
4.9 Effect of bounding the direction within cone. 54
4.10 Error when the direction is not bounded. 54
4.11 Angular deviation within cone. 55
4.12 Angular deviation with respect to globally maximum direction. 55
4.13 Percentage of error for the minimization case. 59
4.14 Angular deviation for the minimization case. 60

5.1 Visibility ratio of line segments. 62
5.2 Mapping the segments to unit circle. 63
5.3 Finding the optimal direction. 63
5.4 Mapping the segments to unit circle. 65
5.5 Finding the optimal direction. 66

List of Tables

4.1 Effect of ordering of line segments. 56
4.2 Position of the optimal direction within a view cone. 57
4.3 Position of the heuristic based direction within a view cone. . 58

Chapter 1

Introduction

A scene is made up of 2D and 3D objects. Projection of objects in 2D and 3D is a well studied problem in computational geometry. Aside from the theoretical interest, its application reaches in the domain of computer graphics [13], object reconstruction [5, 6], machine vision [2], computational geometry [11, 12], and three dimensional graph drawing [8].

Projection involves a view point where our eye or camera is situated, a plane on which the projection is taken and the object of interest. There are two broad classes of projections. The view point, often called the center of projection, may be at infinite distance from the plane of projection producing an orthogonal projection or may be at finite distance from the plane of projection producing perspective projection of the object. Whether it is orthogonal projection or perspective projection, different orientation of the object (or equivalently different position of the view point) produces projections with different characteristics. Like, for some position of the view point a particular set of faces, edges and vertices are visible and for some other position the set of faces, edges and vertices may be completely different. All projections of the same object is therefore not of same. Alternatively, some projections of an object may be more desirable than others.

Given 3D objects such as a set of line segments, triangles or polyhedra it is a well studied problem to compute its "nice" projections based on different criteria for "niceness". In this work, nice projection implies optimal projection having some special geometric property. The term "nice" is a relative term, it actually refers to optimal projection. The criteria for "niceness" depends on various geometric characteristics of the projection of an object. Some of these criteria are more desirable than others depending on the application on mind. For example, it might be more desirable to view a line segment so that its projection does not reduce to a point. Some other common and popular criteria of niceness includes but not limited to finding the maximum and minimum area projections of convex polyhedra, finding minimum crossing projection of 3D line segments, generation of silhouettes of convex polyhedra with certain properties and finding the direction from where the visibility ratio is optimal.

McKenna and Seidel [14] studied the problem of computing maximum and minimum area projection of convex polytopes in R^d. They considered two algorithms, one takes $O(n^{d-1})$ time and space and another takes $O(n^{d-1}\log n)$ time and $O(n)$ space to find the optimal view point, where n is the number of vertices of the polytopes. According to their idea, they divide the d-dimensional space into a set of conical regions which are centered at the origin and correspond to the views of a given polyhedron. Then they cut each conical region by a certain plane and the resulting bounded regions of all cones form a zonotope. They showed that the largest(smallest) shadow of convex polytopes equals the radius of the smallest circumscribed (largest inscribed) sphere of the zonotope. Figure 1.1 shows an example of maximum and minimum area projection of a cuboid.

In a similar problem, Burger and Gritzmann [7] have studied the problem of computing minimum and maximum volume of orthogonal projections of convex polytopes in arbitrary lower dimensions. They have shown that although it might be easy to compute the volume of the projection in a fixed dimension, computing it in arbitrary lower dimension is NP-hard. Then they give several polynomial time approximation algorithms.

Bose *et al.* [4] studied this problem for line segments in 3D. In their algorithms, the criteria for niceness include minimum crossings among line segments, minimum overlapping among line segments and vertices and monotonicity of polygonal chains. For example, in Fig 1.2 (a), there is a single

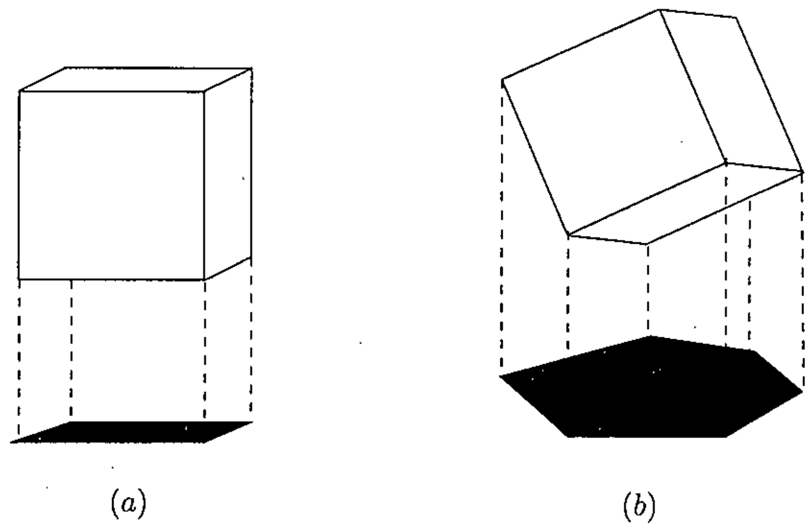


Figure 1.1: (a) Minimum area projection and (b) maximum area projection of a cuboid.

crossing in the projections of two line segments. But in Fig 1.2 (b), for a different view direction, the projections of these line segments does not have any crossing. Eades et al. [8] also studied this problem with similar criteria from the view point of three dimensional graph drawing.

A silhouette is formed by the boundary edges of the projections of convex polyhedra. Recently, Biedl et al. [3] have studied the problem of computing projections of convex polyhedra such that the silhouette (i.e., the projection boundary) meets certain criteria. They have given several algorithms where a given set of vertices, edges and/or faces appear on the silhouette. For example, edges e_1, e_2 and e_3 are on the boundary of all four projections of a polyhedron in Figure 1.3.

Ashraful *et al.* [1] studied the problem of finding orthogonal projections such that within a particular view the minimum visibility ratio over all visible faces (similarly over all visible edges) is maximized where the "visibility ratio" is the ratio of projected area and the actual area of a face. For example, for the visible faces of the polyhedron in Figure 1.4(a), their algorithm generates a projection like that in Figure 1.4(b) as a nice projection. Their algorithms

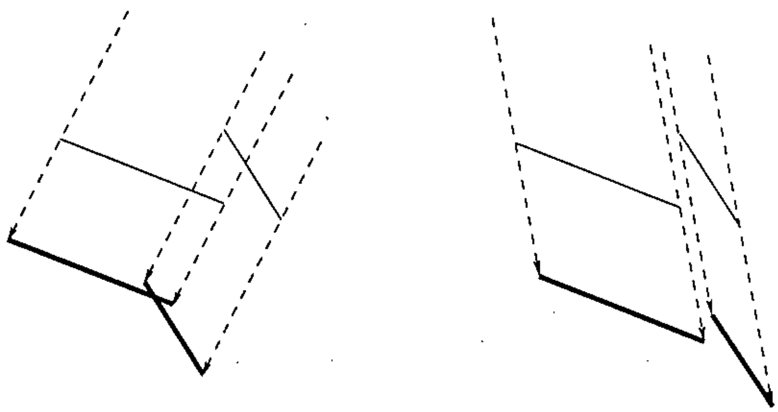


Figure 1.2: (a) Only one crossing in the projections of two line segments and (b) no crossing in the projections for a changed view direction.

also guarantee that no degeneration will occur for visible faces.

1.1 The problems

In this thesis, we study four new criteria of nice projections of 2D and 3D scene that has not been considered so far. These are-

(a) finding the directions of projection in 2D for which the sum of projected lengths of a set of line segments in 2D is maximum and minimum.

(b) finding the directions of projection in 3D for which the sum of projected areas of a set of triangles in 3D is maximum and minimum.

(c) finding the directions of projection in 3D for which the sum of projected lengths of a set of line segments in 3D is maximum and minimum, and

(d) finding the directions of projection in 2D for which the minimum (maximum) visibility ratio of a set of line segments in 2D is maximized (minimized).

Here, we considered orthogonal projections only. We did not consider perspective projections. This is because, all the criteria for optimal projec-

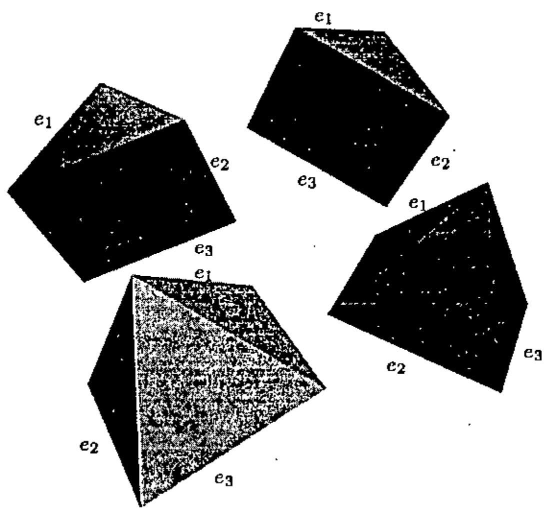


Figure 1.3: The edges e_1, e_2 and e_3 of the polyhedron are visible in the boundary of all the projections.

tions that we have considered in our thesis are not suitable for perspective projections. For example, to maximize the the sum of projections of a set of triangles in 3D, we can place the center of projection on the plane of any triangle and the perspective projection becomes infinity. Similar argument applies for other cases also.

1.2 Outline of the thesis

For solving the first two of our problems, we follow McKenna and Seidel's approach closely. We extend the concept of view from convex polyhedra to 2D and 3D scene and give similar algorithms to solve these problems. For the third problem, we again extend the concept of view differently and give heuristics to find optimal point within a view. We also present some interesting experimental results. At last, we discuss the problem of finding optimal visibility ratio for 2D line segments.

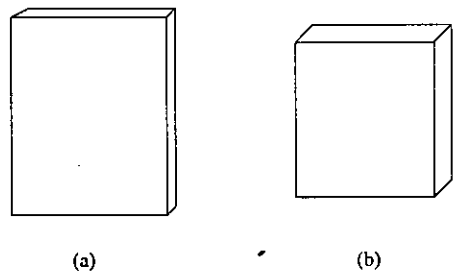


Figure 1.4: (a) A maximum area projection, and (b) a nice projection where the minimum visibility ratio is maximum.

The rest of the thesis is organized as follows. In Chapter 2, we discuss the preliminaries. In Chapter 3, we give algorithms for nice projections of line segments in 2D and triangles in 3D. In Chapter 4, we give heuristic based algorithms to find the approximate direction for optimal projection of line segments in 3D. In Chapter 5, we give algorithms related to optimal visibility ratio of line segments in 2D. Finally, Chapter 6 concludes the thesis with some future work.

Chapter 2

Preliminaries

2.1 Orthogonal and perspective projection

In a planar projection, points are projected onto a plane. Based on the position of the view point (or center of projection), projections can be in general of two types - orthogonal and perspective. In orthogonal projection, the view point is at infinite distance from the plane of projection and is represented as a direction from the view point to the origin. All points are projected in the same direction. In perspective projection, the center of projection is at finite distance. Figure 2.1 shows the two types of projections. In our work, we only consider orthogonal projections.

2.2 Convex Polygon, polyheron and polytope

A *convex polygon* is a region bounded by finite number of line segments called edges such that line segment joining any two points inside the bounded region lies entirely within it. A *convex polyhedron* is the bounded intersection of a finite number of half-spaces. In other words, a polyhedron is convex, if a line segment connecting any of its two points is entirely inside of it, otherwise it is non-convex. See Figure 2.2. The closed surface of a convex polyhedron is made up of planar polygons, called *faces*. The faces meet at line segments, called *edges*, and the edges meet at certain endpoints, called *vertices*. A

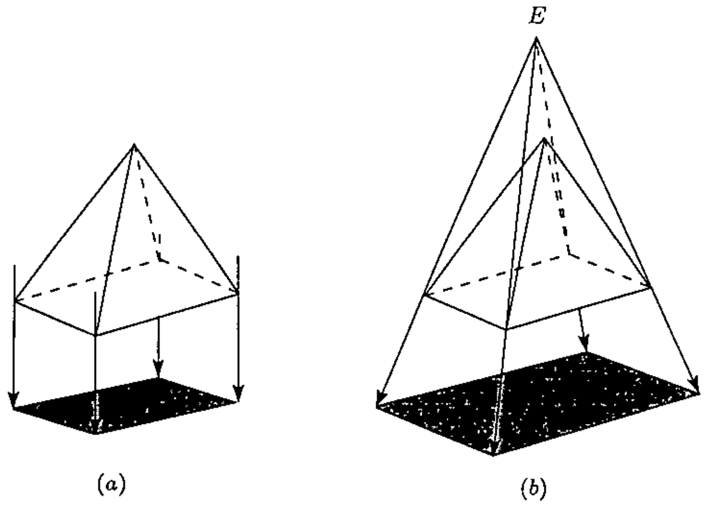


Figure 2.1: (a) An orthogonal projection, and (b) a perspective projection.

convex polytope is a d-dimensional generalization of a 2D convex polygon where $d > 2$.

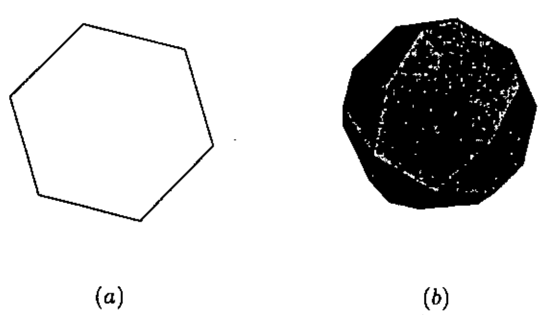


Figure 2.2: (a) A convex polygon, and (b) a convex polyhedron.

2.3 View and view cone

A *normal vector* of line segment l in 2D is the unit vector perpendicular to l. There can be two normal vectors of l which are opposite to each other.

From a given direction d, only one of these two normals are seen. Let us call this visible normal *positive normal* and the other normal *negative normal* . Given a set of line segments in 2D, if we take a line parallel to each segment l_i and translate it to the origin, then they altogether will divide the 2D space and will create a set of conical regions . See Figure 2.3. Inside each cone, a direction sees a different set of normals. We call each of these cones a single *view* of line segment $\{l_i\}$. For example, in Figure 2.3, direction d_1 and d_2 sees different sets of normals and between them the change is only the positive and negative normal of l_1.

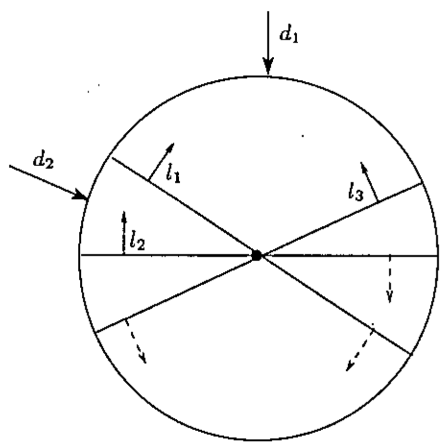


Figure 2.3: Formation in view cone by line segments in 2D. From d_1 to d_2, only l_1's normal changes direction.

Now consider the 3D counterparts of the above concept. A plane π in 3D has exactly two normal vectors which are opposite to each other. From a given direction d in 3D only one of these two normals are seen. Let us call this visible normal *positive normal* and the other normal *negative normal* . Given a set of planes in 3D, for each plane π_i within the set, consider a plane which is parallel to π_i and passes through the origin. Intersection of all such planes divides the 3D space into a set of cones . See Figure 2.4. We call each of these cones a *view cone*. Each view cone represents a single view of the given set, i.e, within a view cone, all the view points will generate the projections in which the set of visible normals remain the same.

For lines in 3D, the concept of view is defined in different approach.

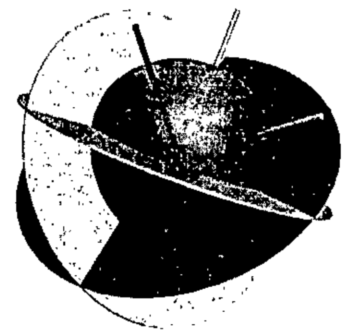


Figure 2.4: Formation of view cone. Left figure shows the case for planes in 3D and right figure shows the case for line segments in 3D.

A line segment l in 3D has infinite many normals, but it has a plane π that is orthogonal to it. Consider a plane that is parallel to π_i and goes through the origin. This plane divides an origin centered unit sphere into two hemespheres. A direction d is either one hemisphere or the other. Let us call this hemisphere (on which d resides) a *positive hemisphere* and the other one a *negative hemisphere* . For a given set of line segments in 3D, all such planes corresponding to the line segments will divide the 3D space into a set of cones. We call these cones *view cones*. Each view cone represents a single view where all the view points lie within the same hemisphere of a line segment.

2.4 Spherical coordinates

A point U in 3D can be defined by spherical coordinates . See Figure 2.5. R is the radial distance of U from the origin O, and ϕ is the angle that U makes with xz-plane, known as *latitude* of U. θ is the *azimuth* of U, the angle between the xy-plane and the plane through U and the y-axis. ϕ lies in the interval $-\pi/2 \leq \phi \leq \pi/2$, and θ lies in the range $0 \leq \theta \leq 2\pi$. With the use of simple trigonometry, it is straightforward to work out the relationships between these quantities and the Cartesian coordinates (u_x, u_y, u_z)

for U. The equations are: $u_x = R\cos\phi\cos\theta, u_y = R\sin\phi, u_z = R\cos\phi\sin\theta$. Spherical coordinates can be used to generate almost all possible vectors in 3D by taking $R = 1$ and varying θ and ϕ within the range by small amount.

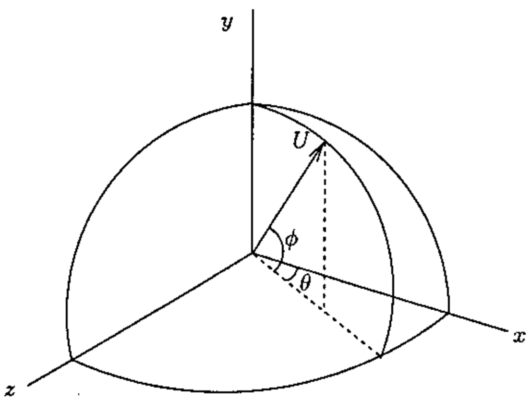


Figure 2.5: Spherical coordinates of U.

2.5 Geodesic distance

Geodesic distance between two points on the surface of a sphere is the minimum distance between them on that surface. In other words, if we draw an arc connecting two points on the surface of the sphere, then its length is the geodesic distance between two points. If we connect these two points with the center of the sphere, we shall get an angle θ which is equivalent to the geodesic distance of these points, provided that the sphere is of unit radius. So, we can always represent the geodesic distance between spherical points by the angle produced by them at the center, and vice versa. See Figure 2.6.

2.6 Visibility ratio

Given a line segment l in 2D and a direction vector d, the projection of l on to a line orthogonal to d is $length(l) \cdot \sin\theta$ where θ is the acute angle between l and d. The *visibility ratio* r_l of a line segment l is the ratio between the

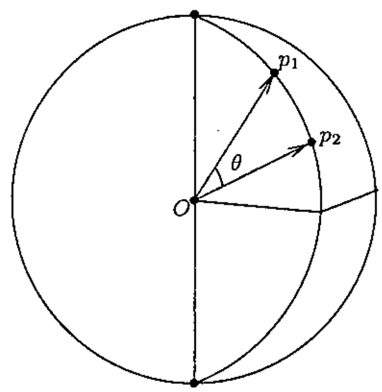


Figure 2.6: Geodesic distance between p_1 and p_2 is equivalent to θ.

projected length and the actual length of l.

$$r_l = \frac{length(l) \cdot \sin\theta}{length(l)} = \sin\theta$$

See Figure 2.7.

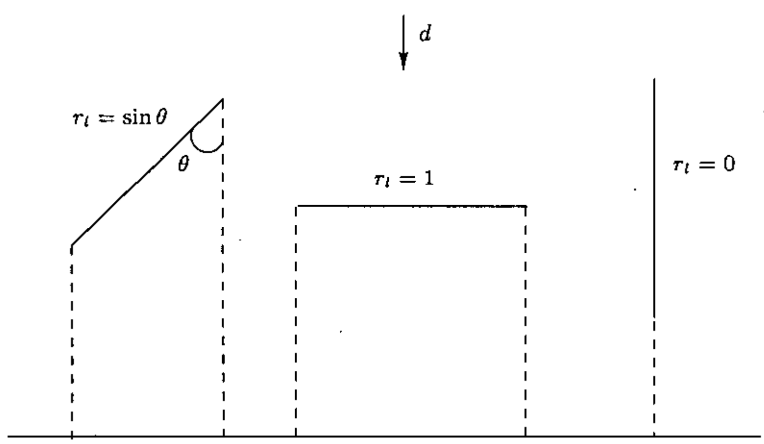


Figure 2.7: Visibility ratio r_l of line segments.

Chapter 3

Optimal projection of 2D and 3D scene

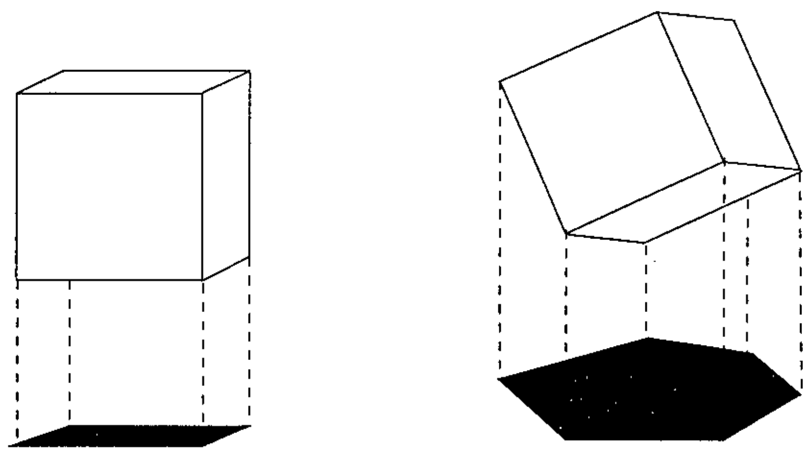
In this Chapter, we present two of our algorithms on nice projections. First, we shall discuss the McKenna and Seidel's algorithm [14] in detail. Later, we shall give algorithms to solve the problems of finding the optimal projection of line segments in 2D and triangles in 3D.

3.1 McKenna and Seidel's approach

McKenna and Seidel studied the problem of placing a light source at infinity so as to maximize or minimize the shadow area of a polytope in R^d. By shadow area they meant the $(d-1)$-volume of the orthogonal projections of the polytope on a hyperplane normal to the direction of illumination. McKenna and Seidel's algorithm can handle polytopes in arbitrary higher dimension. However we will explain their algorithm for 3D for better understanding.

3.1.1 The problem

Given a convex polyhedron P in 3D, the problem is to find the direction for which the area of the orthogonal projections of the polyhedron on a plane normal to the direction is maximum (minimum). For an example, Figure 3.1, the direction of projection is from the top. The left figure shows the minimum area projection and the right one shows the maximum are projection of a cuboid.



(a) (b)

Figure 3.1: (a) Minimum area projection and (b) maximum area projection of a cuboid.

3.1.2 The solution

Let, f be a 2D facet of polyhedron P, and let N_f be the outward normal to facet f with length equal to the area of f. For a facet f, let h_f be the plane that goes through the origin parallel to facet f. A point x is on the positive side of h_f if $x \cdot N_f \geq 0$. For a point x in 3D, let F be $\{\text{facets } f \mid x \text{ is on the positive side of } h_f\}$. F is merely the set of faces visible from direction x at infinity and is called the *visibility set.* Figure 3.2 shows a hexagonal prism of which only 4 faces are visible from the front. These four faces form one such visibility set.

All the h_f's taken together divide 3D space into conical regions. Figure 3.3 shows that all the planes parallel to the visible faces of the polyhedron when translated to the origin, divides the origin centered sphere into conical

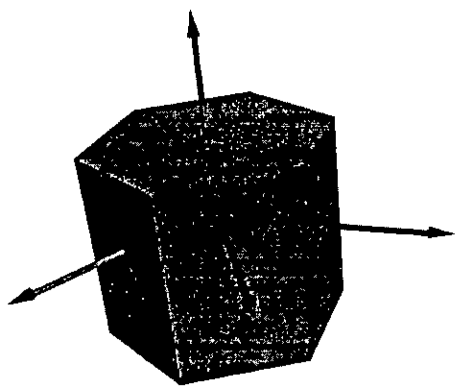


Figure 3.2: Four visible faces form a visibility set.

regions. Therefore, each such cone represents one visible set. For all points in the same cone, the visibility set F is the same. Thus, each cone can be indexed by a set F and thus labeled as C_F. For a particular cone C_F, let, $N_F = \sum_{f \epsilon F} N_f$.

It is to be noted that, for any vector u of unit length and a facet f visible from u the shadow area of f when illuminated from the direction u is $u \cdot N_f$. For any direction u within cone C_F, the shadow area of P when illuminated from direction u is $\sum_{f \epsilon F} u \cdot N_f = u \cdot \sum_{f \epsilon F} N_f = u \cdot N_F$. Thus for an arbitrary point x in cone C_F, $\frac{x}{\mid x \mid} \cdot N_F$ is the area of the shadow when the polyhedron is illuminated from the direction of x, where x is represented as a vector from the origin.

Now, the plane which is perpendicular to N_F, but displaced at a distance of $\frac{1}{\mid N_F \mid}$ away from the origin is defined as $\pi_F = \{x \mid x \cdot N_F = 1\}$. Figure 3.4 shows one such plane. Now it can be stated:

Lemma 3.1.1. *Define B_F to be the interection of cone C_F and plane π_F. If x is a point of B_F then the area of the shadow of polyhedron P when illuminated from the direction of x is $\frac{x}{\mid x \mid} \cdot N_F = \frac{1}{\mid x \mid}$.*

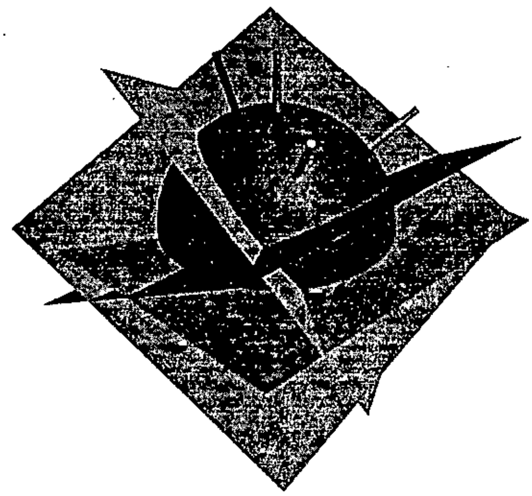


Figure 3.3: The four h_f's divide 3D space into conical regions.

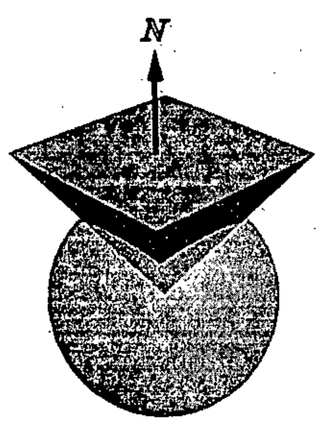


Figure 3.4: The plane, $\pi_F = \{x \mid x \cdot N_F = 1\}$

The above lemma implies that to minimize the shadow area for the illumination direction in C_F one has to find a point v in B_F most distant from the origin. This is easy since B_F is a polygon.

Corollary 3.1.1. *Let v be a vertex of B_F such that $\mid v \mid$ is maximal. For the illumination direction in C_F the minimal shadow area of P is $\frac{1}{\mid v \mid}$ and it is realized by direction v.*

For the shadow area maximization case, the above lemma by itself is not as useful since in general it is not so easy to find the point in B_F closest to the origin and also N_F is not in general contained in B_F. However, for the visibility set for which the shadow area of P is globally maximized, N_F must be in B_F.

Lemma 3.1.2. *Let H_F be the halfspace defined by $\{x \mid x \cdot N_F \leq 1\}$, and let $K_F = C_F \cap H_F$. A point x of R^3 is in the union of all the K_F's if and only if x is in the intersection of all the H_F halfspaces.*

Corollary 3.1.2. *The union of all the K_F's forms a convex polytope Y_p. The facets of Y_p are exactly the B_F's defined earlier.*

Lemma 3.1.3. *Y_p is centrally symmetric about the origin.*

Corollary 3.1.3. *Y_p has a largest inscribed sphere s_p and a smallest circumscribed sphere S_p that are both centered at the origin.*

From the above results, McKenna and Seidel achieve an $O(n^2)$-time algorithm for finding the directions for which the shadow area is maximum (minimum).

Theorem 3.1.1. *Let, R be the radius of the origin centered smallest circumscribed sphere S_p of Y_p. The minimum shadow area of P can be found in $O(n^2)$ time. The minimum shadow area of P is $\frac{1}{R}$ and it is realized for any illumination direction v, where v is a vertex of Y_p that also lies on S_p.*

Theorem 3.1.2. *Let, r be the radius of the origin centered largest inscribed sphere s_p of Y_p. The maximum shadow area of P can be found in $O(n^2)$ time. The maximum shadow area of P is $\frac{1}{r}$ and it is realized for any illumination direction w, where w is any intersection point of s_p and the boundary of Y_p.*

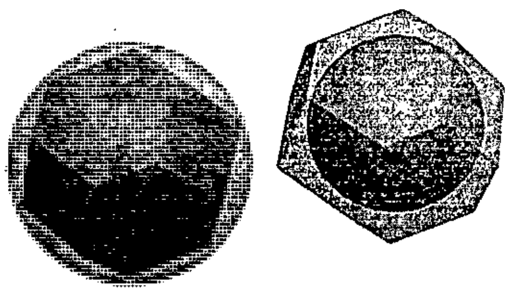


Figure 3.5: The union of all the K_F's forms a convex polyhedron Y_p. Y_p has a smallest circumscribed sphere (left) and a largest inscribed sphere (right) that are both centered at the origin.

3.2 Lines in 2D

McKenna and Seidel worked with polytope in arbitrary dimension. In our work, we used similar idea to solve the problem of finding optimal projection of line segments in 2D and also triangles in 3D. In this section, we describe the problem of finding the optimum projection of line segments in 2D.

3.2.1 The problem

Given n line segments in 2D, the problem is to find a direction vector d for which the sum of projected lengths of the line segments on line perpendicular to d is maximum (minimum). For a particular direction vector d and a line segment l, the length of the projection is $lsin\theta$, where θ is the acute angle between the linesegment l and the direction d. So for n such lines the quantity we want to maximize (minimize) is $\sum_{i=1}^{n} l_i sin\theta_i$.

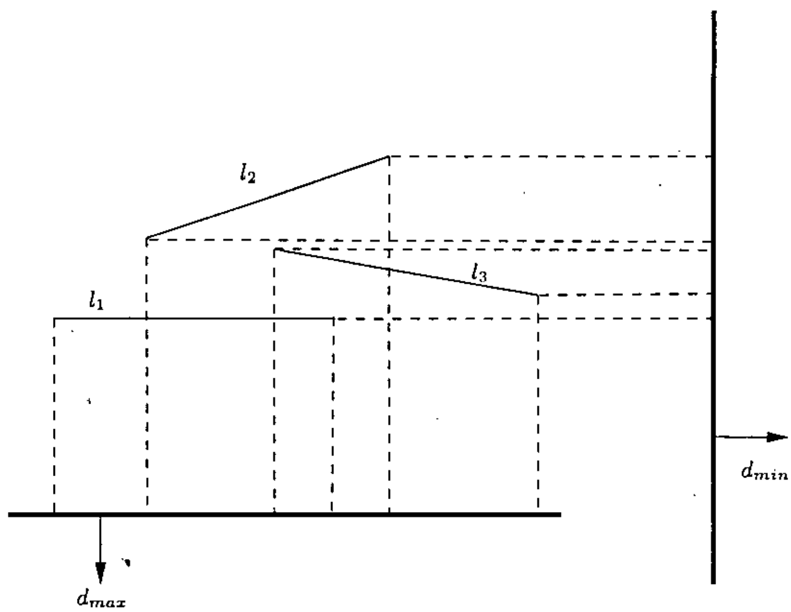


Figure 3.6: The maximum and minimum sum projection of a set of line segments shown by the directions d_{max} and d_{min} respectively.

3.2.2 Concept of view

In McKenna and Seidel, for a particular direction of illumination, they used the concept of view by defining the visibility set. Since they worked with polytopes, it is simply the set of directions from which a particular set of faces of the polytope is visible. But in our case, we are dealing with line segments and for a given direction of illumination we take the sum of projected lengths of *all* line segments. Alternatively, we see all the line segments from all directions and so we do not have any such visibility set. Yet we use the concept of view in our setup by introducing the two opposite normals of each line segment.

Every line segment l_i has exactly two normal vectors: n_{i1} and n_{i2} which are opposite to each other; i.e. $n_{i1} = -n_{i2}$. From the direction d only one of these two normals are seen. Now we can define the set F as $\{n_i \mid n_i.d \geq 0\}$ where n_i is one of the two normal vectors of l_i. This set F is therefore the set of normal vectors that are seen from the direction d. We call it *visible* *normal set.* In Figure 3.7 the solid normals form one such set.

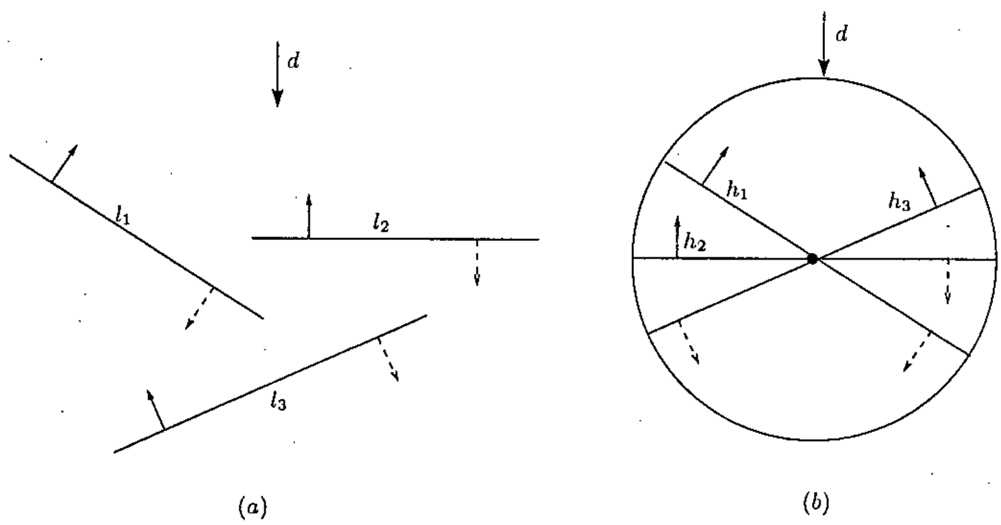


Figure 3.7: (a) Every line segment has two normals. Only one normal (solid) is seen from the direction d, and (b) lines parallel to l_i's divides the origin centered unit circle into conical regions.

3.2.3 Forming the view cone

Let, h_i be the line that goes through the origin parallel to line l_i. All the h_i's taken together divide R^2 into conical regions. Figure 3.7 shows that, all the lines parallel to the line segments when translated to the origin divides the origin centered circle into conical regions. Therefore, each such cone represents one visible normal set. For all points in the same cone, the visible normal set F is the same. Thus, each cone can be indexed by a set F and thus labeled as C_F. For a particular cone C_F, let, $N_F = \sum_{i=1}^{n} n_i$,, where n_i is that normal of l_i for which $n_i \cdot d \geq 0$.

Now, from one such view cone to next view cone exactly one normal vector changes its sign. One thing to be noted that, since we are now working with different normals of the same line in different cones, we can rewrite the quantity we want to maximize (minimize) as $\sum_{i=1}^{n} n_i cos\alpha_i$, where α is the angle between the visible normal n_i and the direction d. Using simple vector notation, it can be written as $\sum_{i=1}^{n} n_i \cdot d$ or $d \cdot \sum_{i=1}^{n} n_i$ or simply $d \cdot N_F$

3.2.4 Finding the optimal point

For any vector u of unit length and a normal n_i visible from u the projected length of l_i when illuminated from the direction u is $u \cdot n_i$. For any direction u within cone C_F, the quantity we want to maximize (minimize) is $d \cdot N_F$. Thus for an arbitrary point x in cone C_F, $\frac{x}{|x|} \cdot N_F$ is the sum of the projected lengths when the direction of projection is x.

Now, the line which is perpendicular to N_F, but displaced at a distance of $\frac{1}{|N_F|}$ away from the origin is defined as $\pi_F = \{x \mid x \cdot N_F = 1\}$. Figure 3.13 shows one such line.

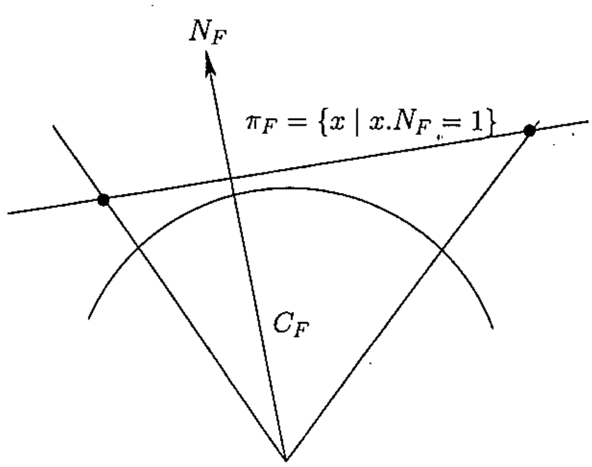


Figure 3.8: The line $\pi_F = \{x \mid x \cdot N_F = 1\}$ is perpendicular to N_F and is at a distance of $\frac{1}{|N_F|}$ away from the origin.

Now it can be stated:

Lemma 3.2.1. *Define B_F to be the interection of cone C_F and line π_F. If x is a point of B_F then the sum of projection from the direction x is* $\frac{x}{|x|} \cdot N_F = \frac{1}{|x|}$

The above lemma implies that, to minimize the sum of projected length for the illumination direction in C_F one has to find a point v in B_F most distant from the origin.

Corollary 3.2.1. *Let, v be a vertex of B_F such that $\mid v \mid$ is maximal. For the illumination direction in C_F the minimal value of the sum of projected lengths is $\frac{1}{\mid v \mid}$ and it is realized by direction v.*

For the maximization case, the above lemma by itself is not as useful since in general it is not so easy to find the point in B_F closest to the origin and also N_F is not in general contained in B_F. However, for the visible normal set for which the quantity is globally maximized, N_F must be in B_F.

The argument is as follows: first, we show that the B_F's taken together form the boundary of a convex polygon Y_p. This polygon is centrally symmetric about the origin and therefore has a maximal inscribed sphere s_p centered at the origin. Any point w on the boundary of Y_p that is closest to the origin must be a point on s_p. The B_F containing w must be tangent to s_p which, since the origin is the center of s_p, implies that w is a multiple of N_F, the normal vector of B_F, i.e. w must be N_F.

Lemma 3.2.2. *Let, H_F be the halfplane defined by $\{x \mid x \cdot N_F \leq 1\}$, and let $K_F = C_F \cap H_F$. A point x of R^2 is in the union of all the K_F's if and only if x is in the intersection of all the H_F halfplanes.*

Proof. (=>) Let, x be a point in the K_F corresponding to some normal vector collection F, i.e. $x \cdot N_F \leq 1$. Now $x \cdot n_i \geq 0$ for all normals n_i in F, and $x \cdot n_i \leq 0$ for all normals n_i not in F. Let, G be any other normal collection. Note that, $G = F \cup (G - F) - (F - G)$, so $x \cdot N_G = x \cdot \sum_{n_i \epsilon G} n_i = x \cdot \sum_{n_i \epsilon F} n_i + x \cdot \sum_{n_i \epsilon G-F} n_i - x \cdot \sum_{n_i \epsilon F-G} n_i$

Point x is in K_F, so the first of the three latter terms is ≤ 1. The summands in the second of these three terms are all negative and summands in the third term are all positive; so the sum of the three terms is ≤ 1. Thus, $x.N_G \leq 1$. Therefore, x is in H_G.

(<=) Let point x lie in all the halfplanes determined by the H_F's. Now x is in some cone C_F; so x lies in $C_F \cap H_F = K_F$. □

Corollary 3.2.2. *The union of all the K_F's forms a convex polygon Y_p. The edges of Y_p are exactly the B_F's defined earlier.*

Now observe that, sum of projected lengths is the same for opposite directions of projection. Thus, $\frac{x}{|x|}.N_F = \frac{-x}{|x|}.N_G$ for all pairs of $(x, -x)$ of points in opposite cones C_F and C_G. Thus $N_F = -N_G$; i.e, the N_F's form a centrally symmetric set about the origin. Therefore $H_G = \{x \mid x \cdot -N_F \leq 1\} = -H_F$ Hence, the intersection of all the different H_F halfplanes forms a convex polygon centrally symmetric about the origin.

Lemma 3.2.3. *Y_p is centrally symmetric about the origin.*

Corollary 3.2.3. *Y_p has a largest inscribed circle s_p and a smallest circumscribed circle S_p that are both centered at the origin.*

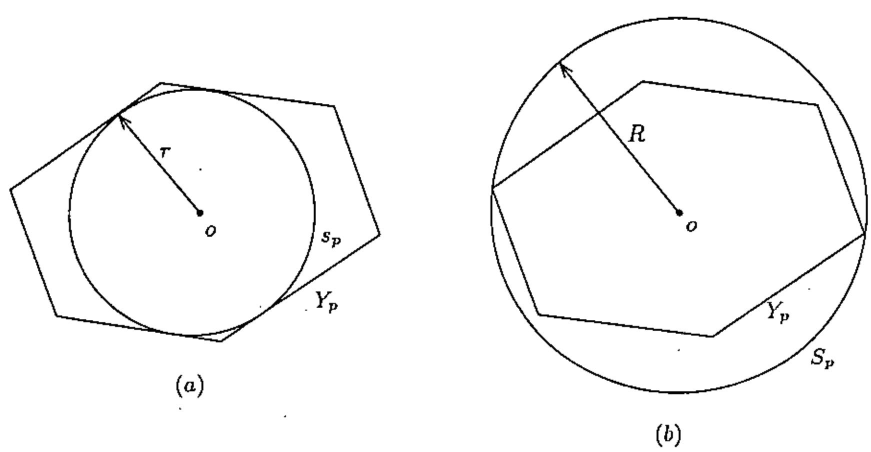


Figure 3.9: The union of all the K_F's forms a convex polygon Y_p. Y_p has (a) largest inscribed circle s_p, and (b) smallest circumscribed circle S_p that are both centered at the origin.

These immediately imply the following two theorems:

Theorem 3.2.1. *Let, R be the radius of the origin centered smallest circumscribed circle S_p of Y_p. The minimum value of the sum of projected lengths can be found in $O(n \log n)$ time complexity and the value is $\frac{1}{R}$ and it is realized for any illumination direction v, where v is a vertex of Y_p that also lies on S_p.*

Proof. Translating the line segments to the origin takes $O(n)$ time. In order to form the cones, we need to sort the line segments based on their polar angles, which takes $O(n \log n)$ time. We associate n_i at each point that we get at the intersection of the origin centered unit circle and l_i's. Calculation of N_F for an arbitrary cone C_F takes $O(n)$ time. Since from one cone to the adjacent cone only one normal changes direction, we can compute N_F for succeeding cones in just $O(1)$ time. While going from one cone to another we keep track of the largest N_F. While computing N_F's in this fashion we can keep track of the longest N_F. The maximum sum of projected lengths is the $|N_F|$ corresponding to the longest N_F and the optimal direction is also the direction of N_F. This step takes $O(n)$ time. Sorting of line segments dominates the time complexity. Overall complexity of our algorithm is thus $O(n \log n)$. □

Theorem 3.2.2. *Let, r be the radius of the origin centered largest inscribed circle s_p of Y_p. The maximum value of the sum of projected lengths can be found in $O(n \log n)$ time complexity and the value is $\frac{1}{r}$ and it is realized for any illumination direction w, where w is any intersection point of s_p and the boundary of Y_p.*

Proof. Translating the line segments to the origin takes $O(n)$ time. In order to form the cones, we need to sort the line segments based on their polar angles, which takes $O(n \log n)$ time. We associate n_i at each point p that we get at the intersection of the origin centered unit circle and l_i's. Calculation of N_F for an arbitrary cone C_F takes $O(n)$ time. Since from one cone to the adjacent cone only one normal changes direction, we can compute N_F for succeeding cones in just $O(1)$. Optimal direction is determined by the point for which $\frac{p}{|p|} \cdot N_F$ is smallest. Checking all $2n$ points we can compute the optimal direction in $O(n)$. Sorting of line segments dominates the time complexity. Overall complexity of our algorithm is thus $O(n \log n)$. □

3.3 Triangles in 3D

In this section, we describe the problem of finding the optimum projection of triangles in 3D. The reason of considering triangles instead other objects is that any 3D object with planar surface can be triangulated. Although in this section we deal with triangles in 3D, the problem if finding the optimum exposure for line segments in 2D (discussed in the previous section) is mathematically similar. We now state the problem formally and describe the solution.

3.3.1 The problem

Given n triangles in 3D, the problem is to find a direction vector d for which the sum of projected area of the triangles on plane perpendicular to d is maximum (minimum). For a particular direction vector d and a triangle t, the projected area of t is $t \cos\theta$, where θ is the angle between the normal to the plane containing t and the direction d. So for n such triangles the quantity we want to maximize (minimize) is $\sum_{i=1}^{n} t_i \cos\theta_i$.

3.3.2 Concept of view and view cone

Like line segments in 2D, we define similar concept of view for the triangles. Every triangle t_i has exactly two normal vectors: n_{i1} and n_{i2} which are opposite to each other; i.e. $n_{i1} = -n_{i2}$. From the direction d only one of these two normals are seen. Now we can define the set, F as $\{n_i \mid n_i \cdot d \geq 0\}$ where n_i is one of the two normal vectors of t_i. This set F is therefore the set of normal vectors that are seen from the direction d. We call it *visible normal set* for the direction d.

Let, h_i be the plane that goes through the origin parallel to the plane containing triangle t_i. All the h_i's taken together divide R^3 into conical regions. Figure 3.12 shows that all the h_i's divides the origin centered unit sphere into conical regions. Therefore, each such cone represents one visible normal set. For all points on the same cone, the visible normal set F is the

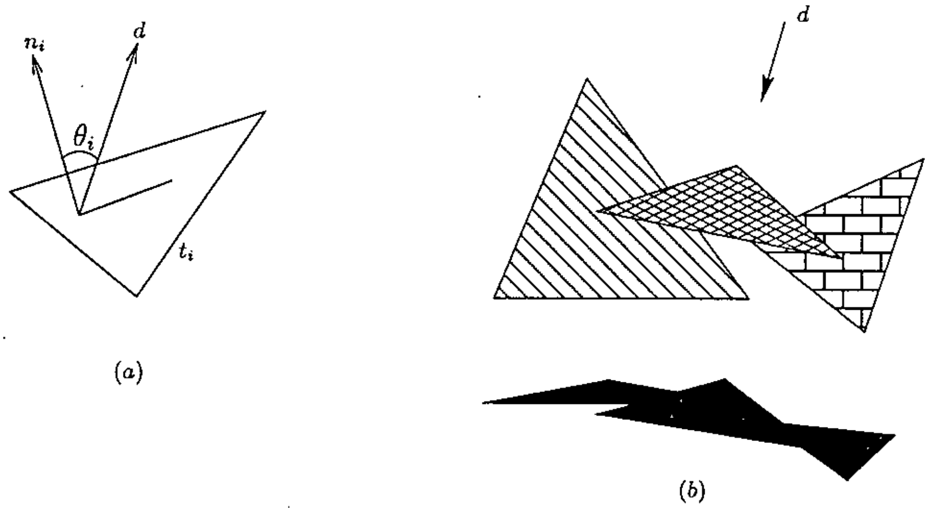


Figure 3.10: (a) θ is the angle between the normal to the plane containing t and the direction d, and (b) projection of triangles on the plane perpendicular to d.

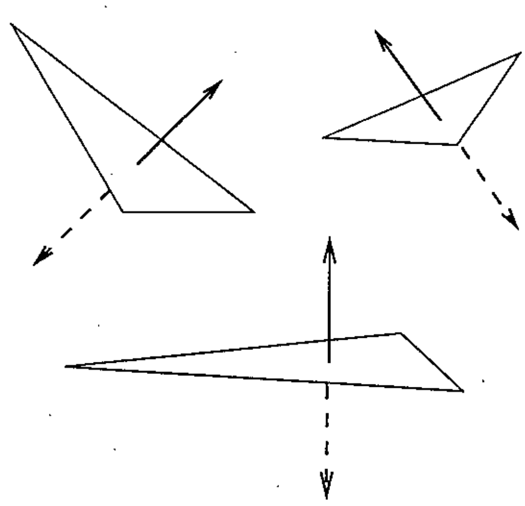


Figure 3.11: Every triangle has two normals. Only one normal (solid) is seen from top and the other normal (dashed) is not seen.

same. Thus each cone can be indexed by a set F and thus labeled as C_F. For a particular cone C_F, let, $N_F = \sum_{i=1}^{n} n_i$, where n_i is that normal of t_i for which $n_i \cdot d \geq 0$.

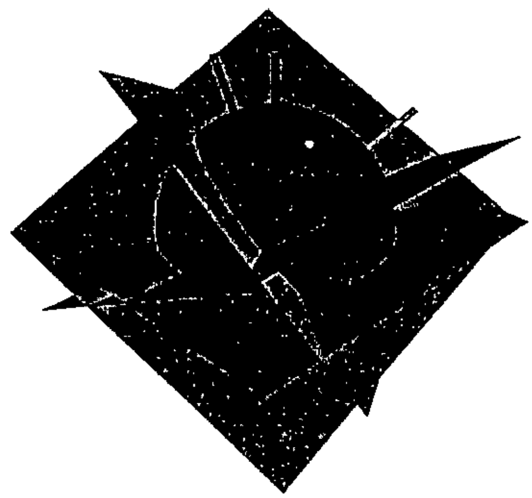


Figure 3.12: All the h_i's divides the origin centered unit sphere into conical regions.

From one view cone to next view cone exactly one normal vector changes its sign. One thing to be noted that, since we are now working with different normals of the same triangle in different cones, we write the quantity we want to maximize (minimize) as $\sum_{i=1}^{n} n_i \cos \alpha_i$, where α_i is the angle between the visible normal n_i and the direction d. Using simple vector notation it can be written as $\sum_{i=1}^{n} n_i \cdot d$ or $d \cdot \sum_{i=1}^{n} n_i$ or simply $d \cdot N_F$

3.3.3 Finding the Optimal point

For any vector u of unit length and a normal n_i visible from u, the projected area of t_i when illuminated from the direction u is $u \cdot n_i$. For any direction u within cone C_F, the quantity we want to maximize (minimize) is $d \cdot N_F$. Thus, for an arbitrary point x in cone C_F, $\frac{x}{|x|} \cdot N_F$ is the sum of the projected areas when the direction of projection is x.

Now, let us define the plane which is perpendicular to N_F, but displaced at a distance of $\frac{1}{|N_F|}$ away from the origin as $\pi_F = \{x \mid x \cdot N_F = 1\}$. Figure 3.13 shows one such plane.

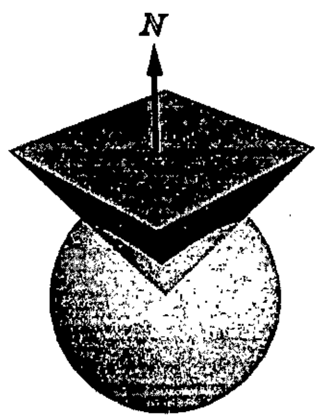


Figure 3.13: The plane $\pi_F = \{x \mid x \cdot N_F = 1\}$ is perpendicular to N_F and is at a distance of $\frac{1}{|N_F|}$ away from the origin.

Now we get the following lemma:

Lemma 3.3.1. *Define B_F to be the interection of cone C_F and plane π_F. If x is a point of B_F, then the sum of projected area from the direction x is* $\frac{x}{|x|} \cdot N_F = \frac{1}{|x|}$.

The above lemma implies that to minimize the sum of projected area for the illumination direction in C_F one has to find a point v in B_F most distant from the origin.

Corollary 3.3.1. *Let, v be a vertex of B_F such that $|v|$ is maximal. For the illumination direction in C_F the minimal value of the sum of projected areas is $\frac{1}{|v|}$ and it is realized by direction v.*

For the maximization case, the above lemma by itself is not as useful since in general it is not so easy to find the point in B_F closest to the origin and also N_F is not in general contained in B_F. However, for the visible normal set for which the quantity is globally maximized, N_F must be in B_F. The argument is as follows: first we show that the B_F's taken together form the boundary of a convex polyhedron Y_p. This polyhedron is centrally symmetric about the origin and therefore has a maximal inscribed sphere s_p centered at the origin. Any point w on the boundary of Y_p that is closest to the origin must be a point on s_p. The B_F containing w must be tangent to s_p which, since the origin is the center of s_p, implies that w is a multiple of N_F, the normal vector of B_F, i.e. w must be N_F.

Lemma 3.3.2. *Let, H_F be the halfspace defined by $\{x \mid x \cdot N_F \leq 1\}$, and let $K_F = C_F \cap H_F$. A point x of R^2 is in the union of all the K_F's if and only if x is in the intersection of all the H_F halfspaces.*

Proof. (=>) Let, x be a point in the K_F corresponding to some normal vector collection F, i.e. $x \cdot N_F \leq 1$. Now $x \cdot n_i \geq 0$ for all normals n_i in F, and $x \cdot n_i \leq 0$ for all normals n_i not in F. Let, G be any other normal collection. Note that, $G = F \cup (G-F) - (F-G)$, so $x \cdot N_G = x \cdot \sum_{n_i \epsilon G} n_i = x \cdot \sum_{n_i \epsilon F} n_i + x \cdot \sum_{n_i \epsilon G-F} n_i - x \cdot \sum_{n_i \epsilon F-G} n_i$. Since point x is in K_F, the first of the three latter terms is < 1. The summands in the second of these three terms are all negative and summands in the third term are all positive; so the sum of the three terms is < 1. Thus $x \cdot N_G \leq 1$. Therefore, x is in H_G.
(<=) Let, point x lie in all the halfplanes determined by the H_F's. Now x is in some cone C_F; so x lies in $C_F \cap H_F = K_F$. □

Corollary 3.3.2. *The union of all the K_F's forms a convex polygon Y_p. The edges of Y_p are exactly the B_F's defined earlier.*

Now observe that sum of projected areas is the same for opposite directions of projection. Thus $\frac{x}{\mid x \mid} \cdot N_F = \frac{-x}{\mid x \mid} \cdot N_G$ for all pairs of $(x, -x)$ of points in opposite cones C_F and C_G. Thus $N_F = -N_G$; i.e, the N_F's form a centrally symmetric set about the origin. Therefore $H_G = \{x \mid x \cdot -N_F \leq 1\} = -H_F$ Hence, the intersection of all the different H_F halfspaces forms a convex polytope centrally symmetric about the origin.

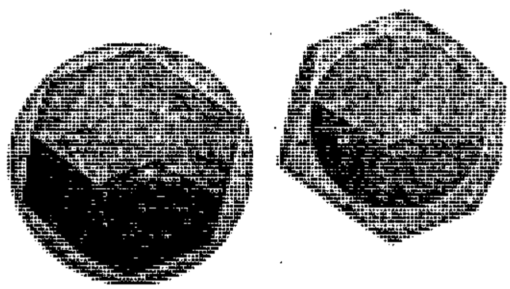


Figure 3.14: The union of all the K_F's forms a convex polyhedron Y_p. Y_p has a smallest circumscribed sphere (left figure) and a largest inscribed sphere (right figure) that are both centered at the origin.

Lemma 3.3.3. *Y_p is centrally symmetric about the origin.*

Corollary 3.3.3. *Y_p has a largest inscribed sphere s_p and a smallest circumscribed sphere S_p that are both centered at the origin.*

These immediately imply the following two theorems:

Theorem 3.3.1. *Let, r be the radius of the origin centered largest inscribed sphere s_p of Y_p. The maximum value of the sum of projected areas can be found in $O(n^2)$ time. The maximum value of the sum of projected areas is $\frac{1}{r}$ and it is realized for any illumination direction w, where w is any intersection point of s_p and the boundary of Y_p.*

Proof. Let, X_1 be the plane $X_1 = \{x \mid x \cdot (1,0,0) = 1\}$. For every plane h_i let $\lambda_i = X_1 \cap h_i$ which is a line on X_1. So for every cone bounded by h_i's there is a polygon in X_1 bounded by λ_i's. Using the algorithm shown in [9] [10] we can construct the graph of all the polygons that results from intersections of λ_i's. Each node of the graph is either a polygon, an edge or a vertex. Each polygon-edge pair and edge-vertex pair is connected by an arc of the graph. Let us call this structure an *arrangement* that takes $O(n^2)$ time to construct. Each node corresponding to an edge is assigned n_i and for each node representing a vertex we keep a pointer that points to the vertex of any incident polygon. Now we can take any polygon of the arrangement and construct the N_F of the cone C_F corresponding to the chosen polygon in $O(n)$. Using the graph, all other nodes corresponding to polygons are visited and we assign them appropriate N_F's. Note that, using the N_F of current C_F we can incrementally compute N_F for the next adjacent polygon in $O(1)$ since two polygons of the arrangement sharing a common edge will have the same N_F except for the difference of the n_i held in the sharing edge. While computing N_F's in this fashion we can keep track of the longest N_F. The maximum sum of projected areas is the $|N_F|$ corresponding to the longest N_F and the optimal direction is also the direction of N_F. Construction of arrangement dominates the time complexity. Thus, the overall complexity of our algorithm is $O(n^2)$. □

Theorem 3.3.2. *Let, R be the radius of the origin centered smallest circumscribed sphere S_p of Y_p. The minimum value of the sum of projected areas can be found in $O(n^2)$ time. The minimum value of the sum of projected areas is $\frac{1}{R}$ and it is realized for any illumination direction v, where v is a vertex of Y_p that also lies on S_p.*

Proof. Let, X_1 be the plane $X_1 = \{x \mid x \cdot (1,0,0) = 1\}$. For every plane h_i let $\lambda_i = X_1 \cap h_i$ which is a line on X_1. So for every cone bounded by h_i's there is a polygon in X_1 bounded by λ_i's. Using the algorithm shown in [9] [10] we can construct the graph of all the polygons that results from intersections of λ_i's. Each node of the graph is either a polygon, an edge or a vertex. Each polygon-edge pair and edge-vertex pair is connected by an arc of the graph. Let us call this structure an *arrangement* that takes $O(n^2)$ time to construct. Each node corresponding to an edge is assigned n_i and for each node representing a vertex we keep a pointer that points to the vertex of any incident polygon. Now we can take any polygon of the arrangement and construct the N_F of the cone C_F corresponding to the chosen polygon in $O(n)$. Using the graph, all other nodes corresponding to polygons are visited and we assign them appropriate N_F's. Note that, using the N_F of current C_F we can incrementally compute N_F for the next adjacent polygon in $O(1)$ since two polygons of the arrangement sharing a common edge will have the same N_F except for the difference of the n_i held in the sharing edge. For each vertex v of Y_p, there exists a vertex v' of the arrangement in X_1 that has the same direction as v. Exploring the vertices in the arrangement graph and keeping record of the smallest $\frac{v'}{|v'|} \cdot N_F$ the optimal direction is found. Construction of arrangement is the dominating step. Thus, the overall complexity of our algorithm is $O(n^2)$. □

Chapter 4

Lines in 3D

In this chapter, we present our heuristic based algorithm on nice projection of line segments in 3D . First, we shall describe the problem. Then we shall explain our heuristic and give the algorithm in detail. Later, we shall show the experimental results.

4.1 The problem

Given n line segments in 3D, the problem is to find a direction vector d for which the sum of projected lengths of the line segments on the plane perpendicular to d is maximum (minimum). For a direction vector d and a line segment l, the length of the projection is $l sin\theta$, where θ is the acute angle between the line segment l and the direction d. So for n such lines the quantity we want to maximize (minimize) is $\sum_{i=1}^{n} l_i sin\theta_i$. Using vector notation it can be written as $\sum_{i=1}^{n} |l_i \times d|$.

4.2 The solution

At first glance, the formal definition of the problem seems similar to the problems we have described in Chapter 3. Although they are similar in nature, this problem is more difficult as it deals with line segments in 3D.

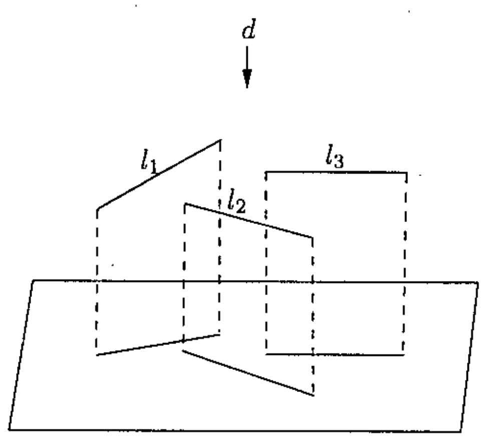


Figure 4.1: Projection of line segments in 3D.

Figure 4.2 shows the plot of $\sum_{i=1}^{n} |l_i \times d|$ for all possible directions. In this figure, there are only 10 line segments, yet we see that there are multiple local maxima and minima on the curve. It is to be noted that for line segments in 2D (and triangles in 3D) we restated the quantity to optimize from $\sum_{i=1}^{n} l_i sin\theta_i$ to $\sum_{i=1}^{n} n_i cos\alpha_i$, by introducing the concept of visible normal n_i from the direction d. We then expressed the expression using dot product of vectors and solved the problem analytically. But for this problem, we cannot do this since a line segment in 3D has infinite number of directions that are orthogonal to it. So, the expression remains as it is and we could not find an straight forward analytical solution to this. We instead give a heuristic based iterative algorithm for this problem.

4.2.1 Concept of view

For line segments in 3D, we cannot find two opposite normals that could be used to define the concept of view as we have done in previous chapter. Instead, we see that, for each line segment l_i we can draw a plane that is orthogonal to the line segment. Figure 4.3 shows the such planes for a line segment and also for a set of line segments. We now define plane π_i that is parallel to this plane and passes through the origin. An origin centered unit sphere S intersects π_i and we get a great circle $c_i = \pi_i \cap S$. π_i essentially

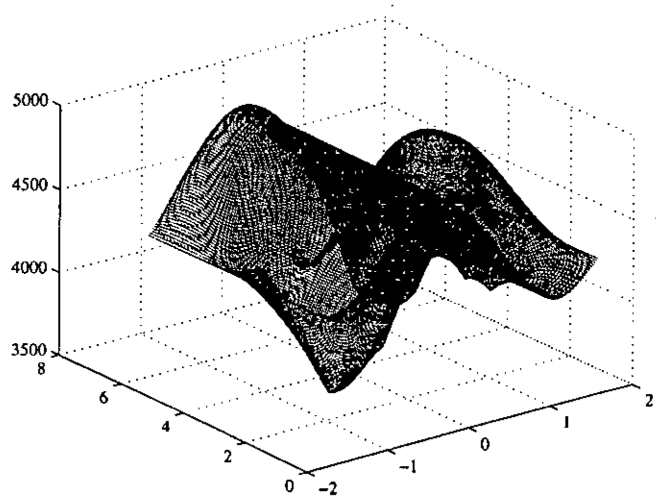


Figure 4.2: The plot of $\sum_{i=1}^{n} |l_i \times d|$ for all possible directions.

divides S into two hemispheres h_{i1} and h_{i2}. Any direction from a point on the perimeter of c_i towards the center of c_i sees li to its greatest length. For any other point $x \in (S - c_i)$ the observer sees l_i shorter. A point $x \in S$ is either on the perimeter of c_i or on h_{i1} or on h_{i2}. Including c_i into any of these two hemispheres, we can, in general, comment that l_i is seen from either h_{i1} or h_{i2}. Now, for all n line segments, all the π_i's will divide S into a number of conical regions. From any point x on any such conical regions C_H, we see each l_i from exactly one of its h_i's. Let us call this hemisphere a positive hemisphere for l_i and let us call the other one a negative hemisphere for l_i. Each C_H, therefore, serves the purpose of a view. We now search for the optimal point in each such C_H. Let us define S_{CH} as the spherical portion of C_H that is bounded by the π_i's.

4.2.2 The heuristic for maximization case

In this section, we describe our algorithm for maximization problem. Our algorithm is heuristic based. Since there are infinite number of normal vectors to a line segment, we give a heuristic to choose one that closely approximates the actual direction. We now describe the trivial cases at first and then

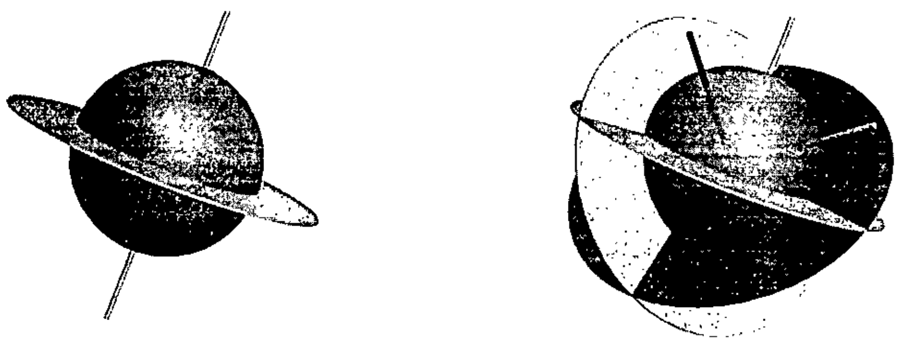


Figure 4.3: Planes parallel to line segments. A single line segment is shown on the left and a set of line segments are shown to its right.

discuss our iterative approach.

The trivial cases

When $n < 2$ we can state the following:

For $n = 1$, i.e. when there is only one line segment l_1, from any point $x \in c_1$, l_1 is seen with its maximum length which is l_1.

For $n = 2$, i.e. when there are exactly two line segments: l_1, l_2, the direction d from which these are seen largest is the direction perpendicular to the plane of l_1 and l_2. The sum of projected lengths of l_1 and l_2 is $l_1 + l_2$

The general case

For $n > 2$, we adopt an incremental approach. Suppose, we have already found the approximate best direction d_{n-1} for $(n-1)$- line segments and now we are presented with the n-th line segment l_n. We give heuristic is to combine d_{n-1} with l_n to find d_n.

For single line segment we know that, in order to see l_n at its maximum length, any point x on the perimeter of the corresponding great circle c_n

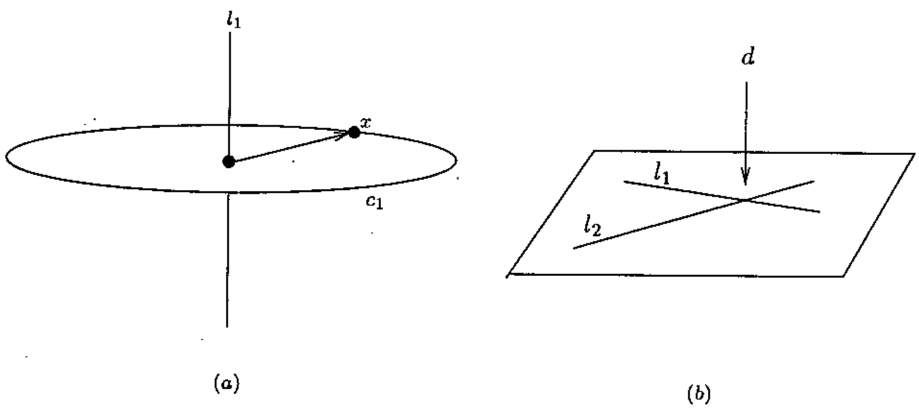


Figure 4.4: (a) For a single line segment, there are infinite many orthogonal directions, and (b) for two line segments, the direction of the cross product is the optimal one.

is the direction. And we already have the approximate best direction d_{n-1} for the previous $(n-1)$ line segments. So, naturally d_n should be as close as possible to the direction d_{n-1} and also at the same time it should lie as close as possible to the perimeter of c_n. For this, we find the point x_p on c_n that is geodesically closest to d_{n-1}. Figure 4.5 shows that point x_p lies on a line that is the intersection of two planes: one is π_n and the other is the plane containing l_n and d_{n-1}. Let us denote the plane containing l_n and d_{n-1} as $\pi_{(l_n,d_{n-1})}$. Now l_n is the normal to π_n and a normal to $\pi_{(l_n,d_{n-1})}$ is $(d_{n-1} \times l_n)$. Since both π_n and $\pi_{(l_n,d_{n-1})}$ pass through the origin, the cross product of their normals determine their intersecting line. The intersection line is therefore $l_n \times (d_{n-1} \times l_n)$. Now x_p being at unit distance from origin $x_p = \dfrac{l_n \times (d_{n-1} \times l_n)}{|l_n \times (d_{n-1} \times l_n)|}$.

Once we have got x_p, we compute d_n as the weighted summation of directions denoted by x_p and d_{n-1}. d_{n-1} is weighted by an amount equal to the sum of projected lengths of $(n-1)$ line segments when the direction of projection is d_{n-1}. And x_p is weighted by the amount equal to the projected length of l_n when seen from x_p. Hence, we get $d_n = (\sum_{i=1}^{n-1} |l_i \times d_{n-1}|)d_{n-1} + |l_n|x_p$.

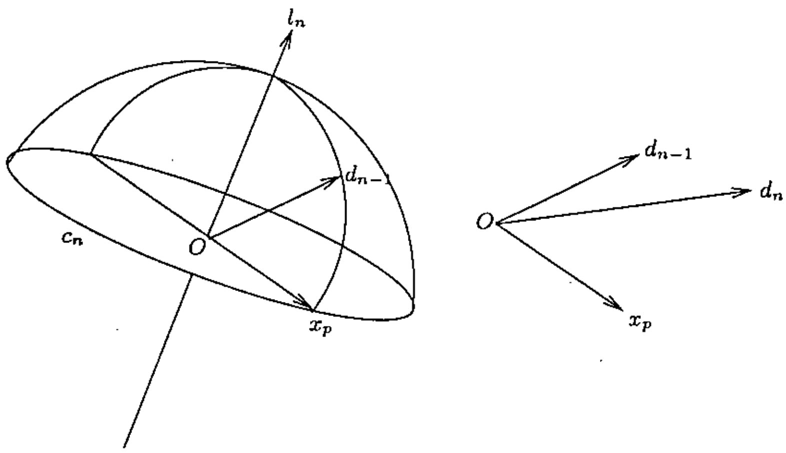


Figure 4.5: Heuristic to determining d_n.

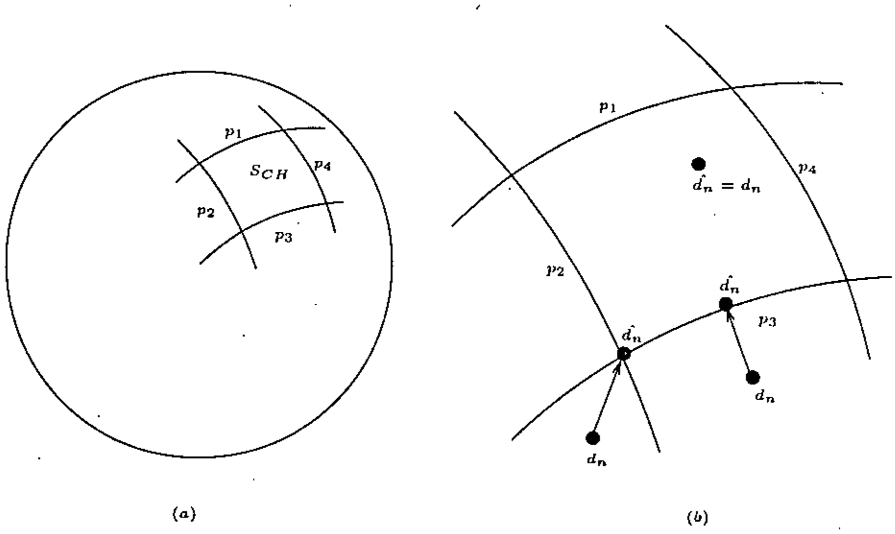


Figure 4.6: (a) Four planes forming the S_{CH}, and (b) three different cases are shown while bounding d_n to $\hat{d_n}$.

Bounding the direction

We have just described the heuristic which must be applied to all the views (i.e. to all the 3D cones C_H). But point to be noted that, in general the direction d_n calculated thus is not always within C_H. So, we need to bound d_n inside the view cone C_H when it is not.

To do this, we at first find S_{CH}. It is merely a spherical polygonal area with maximum n vertices. Now, to bound d_n within S_{CH}, we find the point on the boundary of S_{CH} that is geodesically closest to d_n. For this, we take the projection of d_n on only those p_i's for which d_n is on the negative hemisphere. When the projected point is on any of the edges of S_{CH} then we are done; otherwise we take the corner point of S_{CH} which is closest to d_n. We denote this bounded d_n as $\hat{d}_n$.

Time complexity

For each line segment l_n, we at first compute x_p in $O(1)$ and then compute d_n. Computation of d_n involves $(n-1)$ vector products each taking $O(1)$. So, computing d_n takes $O(n)$ time. Considering all n line segments, the overall complexity of our algorithm for a given view is $O(n^2)$.

4.2.3 The heuristic for minimization case

In this section, we describe our algorithm for minimization problem. Here once again we adopt an incremental approach. But this time, instead of considering each view cone separately, we calculate the approximate direction globally.

The trivial cases

For $n = 1$, i.e. when there is only one line segment l_1, the direction for which l_1's projection is minimum is the direction parallel to l_1 and the projected length is 0.

For $n = 2$, i.e. when there are exactly two line segments, we at first show that the optimal direction from which the summation of projected lengths is minimized lies on the plane containing the line segments. The argument follows. Suppose, $\pi_{(l_1,l_2)}$ is the plane that is parallel to the plane containing l_1 and l_2 and passes through the origin. An origin centered unit sphere S intersects $\pi_{(l_1,l_2)}$ and we get a great circle $c = \pi_{(l_1,l_2)} \cap S$. $\pi_{(l_1,l_2)}$ essentially divides S into two hemispheres h_1 and h_2. For any point x on the surface of the sphere, the sum of projected lengths is $l_1 \sin\theta_{(1,x)} + l_2 \sin\theta_{(2,x)}$ where $\theta_{(1,x)}$ and $\theta_{(2,x)}$ are the acute angles that l_1 and l_2 make with ox. Now for every such x we can define a point x_c that lies on c and geodesically closest to c from x. Let $\theta_{(1,x_c)}$ and $\theta_{(2,x_c)}$ are the acute angles that l_1 and l_2 makes with direction ox_c. Note that ox_c is the projection of ox on the plane $\pi_{(l_1,l_2)}$. Thus $\theta_{(1,x_c)} \leq \theta_{(1,x)}$ and $\theta_{(2,x_c)} \leq \theta_{(2,x)}$. Therefore $l_1 \sin\theta_{(1,x_c)} + l_2 \sin\theta_{(2,x_c)} \leq l_1 \sin\theta_{(1,x)} + l_2 \sin\theta_{(2,x)}$. So we get a smaller sum of projected lengths at x_c. Hence the optimal point is on c. The problem (for $n = 2$) now is reduced to 2D for which we already have given an exact solution in Section 3.2. Using that for $n = 2$ we conclude that the direction of the larger line segment is the optimal direction.

The general case

Being inspired by the trivial cases, we give a simple heuristic to find the direction d_n that minimizes the sum of projected lengths of n line segments in 3D. We hypothesize that, the optimal direction d_n will be the same as the direction of any of the n line segments. We, therefore, take each of the n line segments l_k in turn and compute the value $f(k) = \sum_{i=1}^{n} |l_i \times l_k|$. The direction d is therefore the the direction of l_k for which $f(k)$ is minimum.

Time complexity

For each line segment l_k where $1 \leq k \leq n$, we compute $f(k)$. Computation of $f(k)$ takes $O(n)$ times since we have to perform n vector products each taking $O(1)$ time. Overall time complexity is therefore $O(n^2)$.

4.2.4 Experimental results for maximization problem

We have conducted several experiments. All the experiments were run in a PC having Intel Core 2 duo processor and 2GB RAM. Experimental results shown in the figures and tables are the average values of several independent runs. Input to the programs are all generated at random. In this section, we summarize the results for the maximization problem.

Finding optimal solution using brute force

Since there is no known algorithm to find the optimal solution for this problem, we at first compute it using a brute force solution. Let, C be an origin centered sphere of unit radius. A point U on the spherical surface of C corresponds to a direction which is from the origin O to U. Considering all such U on the surface of C we can generate every possible direction in 3D. Although, U is in 3D, it can be expressed using only two variables if we consider spherical coordinates. We have seen in Chapter 2, how a point U is defined in spherical coordinates. ϕ is the angle that U makes with xz-plane, known as *latitude* of U. θ is the *azimuth* of U, the angle between the xy-plane and the plane through U and the y-axis. ϕ lies in the interval $-\pi/2 \leq \phi \leq \pi/2$, and θ lies in the range $0 \leq \theta \leq 2\pi$. With the use of simple trigonometry, it is straightforward to work out the relationships between these quantities and the Cartesian coordinates (u_x, u_y, u_z) for U. The equations are: $u_x = \cos\phi\cos\theta, u_y = \sin\phi, u_z = \cos\phi\sin\theta$. So varying ϕ within the range $-\pi/2 \leq \phi \leq \pi/2$ by a small amount $\delta\phi$ and varying θ within the range $0 \leq \theta \leq 2\pi$ by a small amount $\delta\theta$ we can generate 3D vectors in almost every possible direction.

In our problem, since we are searching for the optimal point within a view cone, we therefore choose only those U that are within the view cone of our interest. It is easy to check whether U is within the cone or not, as for a particular cone C_H we know exactly which planes p_i form the view and exactly from which side of the plane, l_i is being seen within C_H. So before evaluating the expression we just test for U whether it falls within C_H.So for each such direction $U(u_x, u_y, u_z)$ we calculate the value $\sum_{i=1}^{n} |l_i \times U|$ and take the direction that maximizes the expression.

Accuracy of our algorithm within a bounded cone

In order to verify the accuracy of our algorithm, we compare it with the solution obtained using brute force. Figure 4.7 shows the comparison of the expression value by our algorithm with that of brute force within a particular view cone. In the figure, we see the percentage of error of the heuristic based algorithm's output with respect to the optimal value for various numbers of line segments N. There is a gradual decrease in error as N increases. The reason is that, with larger values of N the spherical polygonal area S_{CH} becomes smaller and hence there is less chance for our algorithm's output to make larger errors. In average percentage of error is small (just 1.48%). Figure 4.8 shows the angular deviation (i.e. the angle between the optimal direction and $\hat{d}_n$) for the same set of line segments. Average deviation is small (just 9.89^0). Both figures agree, as for smaller percentage of error the angular deviation is smaller. It justifies the fact that, our algorithm in deed finds a good approximate direction; not merely a direction that is way distant from the optimal direction yet showing small error.

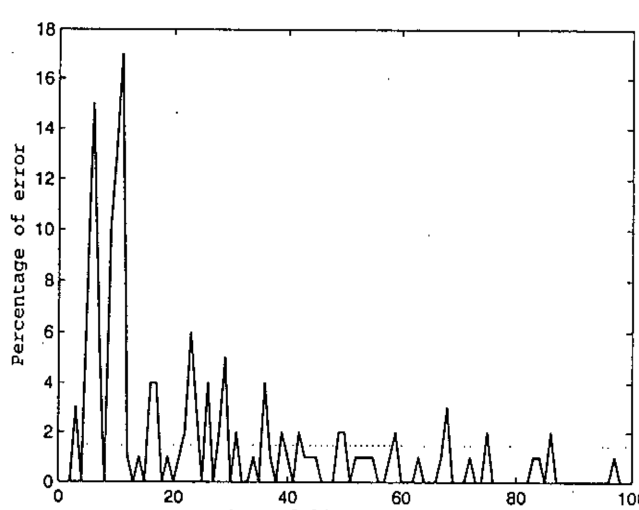


Figure 4.7: Accuracy of our algorithm within a bounded cone.

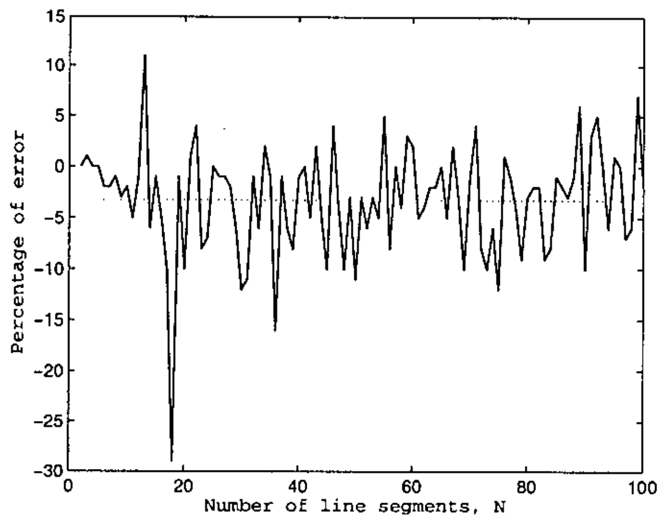


Figure 4.9: Percentage of error when d_n is not bounded with respect to maximal value within cone.

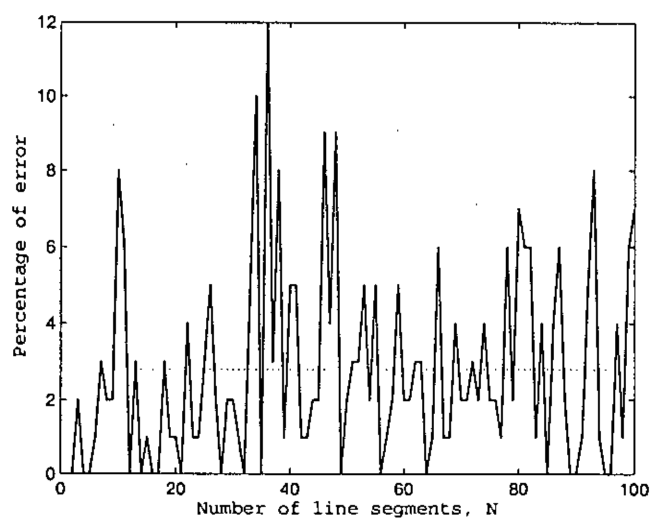


Figure 4.10: Percentage of error when d_n is not bounded with respect to globally maximum value.

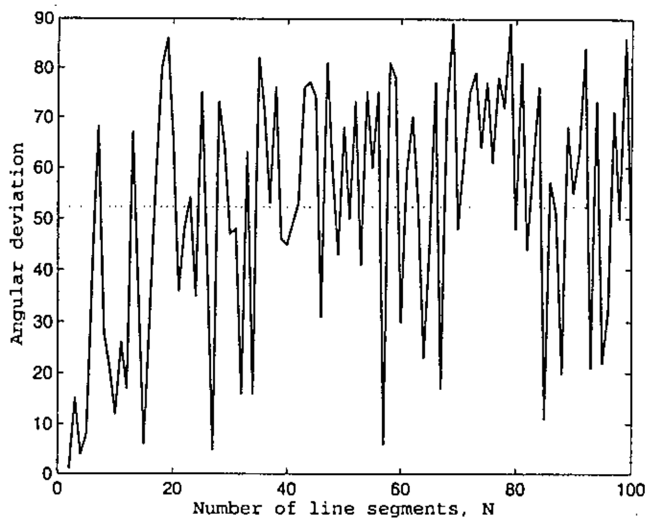


Figure 4.11: Angular deviation with respect to maximal direction within cone when d_n is not bounded to $\hat{d}_n$.

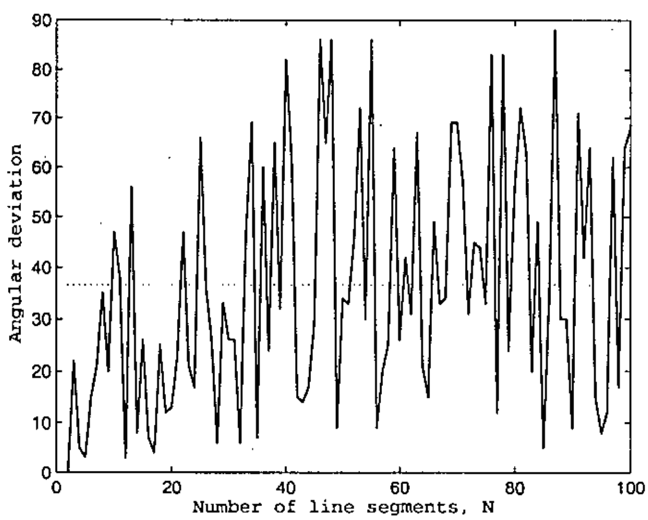


Figure 4.12: Angular deviation with respect to globally maximum direction when d_n is not bounded to $\hat{d}_n$.

Effect of ordering of the line segments

Since our algorithm is incremental and we are unable to find the exact optimal solution, our solution is sensitive to the order of the line segments. To check this, we have applied our algorithm on a set of line segments varying the order of the line segments. Table 4.1 shows that the percentage of error as well as the angular deviation changes for the same set of line segments when the line segments are presented in increasing order, decreasing order and random order of their lengths.

N	Increasing		Decreasing		Random	
	%error	angle	%error	angle	%error	angle
2	0	0	0	0	0	0
5	30	79	1	13	18	48
7	5	34	4	35	4	34
10	0	1	0	1	3	10
12	0	0	1	3	0	3
15	2	12	0	4	9	38
20	0	11	0	8	0	14
25	2	19	12	54	12	54
30	3	16	2	13	0	0
50	0	0	2	22	0	0
60	2	22	1	14	0	0
75	0	1	0	1	0	0
100	1	5	1	6	0	0
200	0	3	0	3	0	3
500	0	3	0	2	0	3
750	0	0	0	0	0	1
1000	0	0	0	0	0	0

Table 4.1: For the same set of line segments the error depends on their ordering.

Position of optimal point within a view cone

We have also conducted experiments to see where the optimal point lies within a view cone. Table 4.2 summarizes the result. For varying numbers of line segments, column 2 shows the number of planes that actually form the view cone of our interest. Column 3 shows the distances of each of these planes. We do not see exact zero distance from any of the planes although for some values of N (like- 5, 20, 100, 750, 1000) it seems that the optimal point is very close to a plane and even for some N (like- 10, 75)it seems very close to a vertex of S_{CH}. For other values of N the optimal point seems rather well inside S_{CH}.

N	Number of planes	Minimum distances from the planes
2	2	{0.000390, 0.000466}
5	4	{0.000203, 0.009113, 0.011231, 0.582377}
10	5	{0.000318, 0.000753, 0.266885, 0.457950, 0.643922}
15	4	{0.000646, 0.001887, 0.099063, 0.115297}
20	4	{0.000189, 0.001294, 0.131297, 0.189344 }
25	5	{0.002149, 0.002188, 0.095061, 0.110113, 0.131350 }
30	5	{0.000392, 0.012080, 0.013770, 0.017340, 0.151812 }
50	3	{0.000910, 0.002890, 0.037029 }
75	3	{0.000120, 0.000343, 0.062127 }
100	4	{0.000020, 0.009598, 0.022386, 0.109364 }
200	5	{0.000546, 0.003377, 0.017035, 0.034460, 0.060357 }
750	5	{0.000409, 0.002131, 0.002316, 0.003228, 0.012894 }
1000	4	{0.000259, 0.000858, 0.001025, 0.001799 }

Table 4.2: Distance of the optimal point from each plane forming the view cone.

We have conducted this same experiment for our $\hat{d}$ also. Our heuristic outputs $\hat{d}$ that is sometimes inside, sometimes on the edge and sometimes on a vertex of S_{CH}. Table 4.3 summarizes the result. The distances are sorted in increasing order. For some N (like- 2, 15, 20 etc) first two distances are zero; this means that $\hat{d}$ is on a vertex of S_{CH}. For some N (like- 5, 20) only the first distance is zero; this means that $\hat{d}$ is on an edge of S_{CH}.

4.2.5 Experimental results for minimization problem

We have conducted several experiments for the minimization problem also. All the experiments were run in a PC having Intel Core 2 duo processor

N	Number of planes	Minimum distances from the planes
2	2	{0.000000, 0.000000}
5	4	{0.000000, 0.040126, 0.341487, 0.536006 }
10	5	{0.003250, 0.153620, 0.201364, 0.347828, 0.525232 }
15	4	{0.000000, 0.000000, 0.098657, 0.116629 }
20	4	{0.000000, 0.004000, 0.030390, 0.191016 }
25	5	{0.000000, 0.000000, 0.044029, 0.206483, 0.373161 }
30	5	{0.000000, 0.000000, 0.052567, 0.073266, 0.138332 }
50	3	{0.000000, 0.000000, 0.041413 }
75	3	{0.000000, 0.000000, 0.126531 }
100	4	{0.000000, 0.000000, 0.043050, 0.106392 }
200	5	{0.000000, 0.000000, 0.018908, 0.023165, 0.071857 }
750	5	{0.000000, 0.000000, 0.008202, 0.009178, 0.015052 }
1000	4	{0.000000, 0.000000, 0.000315, 0.003253}

Table 4.3: Distance of $\hat{d}$ from each plane forming the view cone.

and 2GB RAM. Experimental results shown in the figures and tables are the average values of several independent runs. Input to the programs are all generated at random. Below we summarize them all.

Finding optimal solution using brute force

Like maximization problem, we at first compute the optimal solution using brute force. Let C be an origin centered sphere of unit radius. A point U on the surface of C corresponds to a direction from the origin O to U. Considering all such U on the surface of C we can generate every possible direction in 3D. Here again we express U in spherical coordinates and then find the optimal direction and later we compare it with our solution.

Accuracy of our algorithm

In order to verify the accuracy of our algorithm, we compare it with the solution obtained using brute force. Figure 4.13 and Figure 4.14 shows the comparison of the expression value by our algorithm with that of brute force. In Figure 4.13 we see the percentage of error of the heuristic based algorithm's output with respect to the optimal value for various numbers of line segments N. The average percentage of error is very small (just 0.14%). The error is surprisingly small for most of the values of N. Rather in some cases the error is negative. It is due to the cause that, we could not generate all possible direction in brute force. If we could vary ϕ within the range $-\pi/2 \leq \phi \leq \pi/2$ by an infinitesimally small amount $\delta\phi$ and vary θ within the range $0 \leq \theta \leq 2\pi$ by a infinitesimally small amount $\delta\theta$ we could generate 3D vectors in almost every possible directions and in that case the error would not have been negative. But the important thing is that yet the average error will be very small and we have a very good approximation of the optimal direction. Figure 4.14 shows the angular deviation (i.e. the angle between the optimal direction and d_n) for the same set of line segments. Average deviation is small (just 1.68^0), conforming to our earlier claim of having a very good approximate solution.

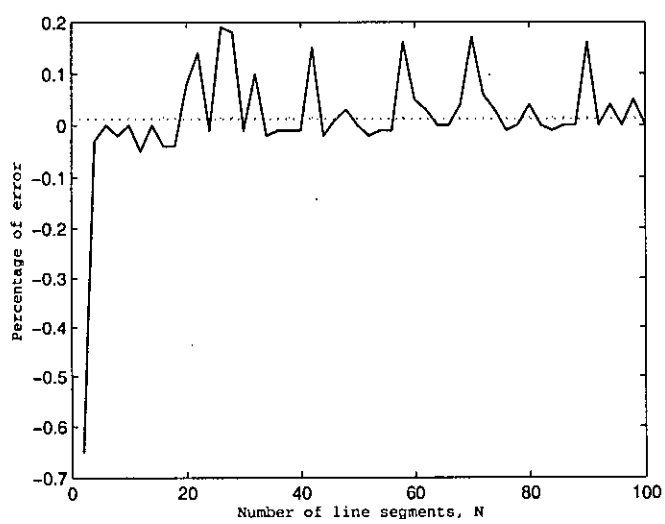


Figure 4.13: The percentage of error.

4.3 A novel application

The above algorithms can be used in a novel application, which is to find the maximum (minimum) perimeter of a convex polyhedron in an orthogonal

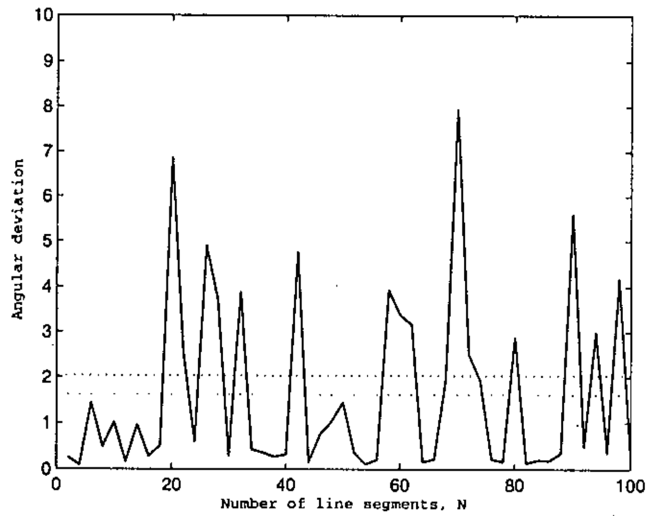


Figure 4.14: The angular deviation.

projection and for which no solution is known. As described in Chapter 2, the faces of a convex polyhedron meet at line segments, called edges. For a given direction, only a set of faces of the polyhedron is visible. Considering all possible directions, there can be finite number of different sets, each of which sets corresponds to a view. In each view, the visible faces are the same and so the visible edges are also the same. Projections of these visible edges forms the perimeter in an orthogonal projection of the polyhedron when viewed from that view. Given a polyhedron, algorithms exist for finding the set of visible edges of a view. We can consider each of the visible edges as a line segment and using our algorithm we can find the approximate direction for which the sum of projection of these edges is maximum (minimum). This direction is the direction of maximum (minimum) perimeter of the polyhedron in an orthogonal projection within a view. Repeating the process for all views of the polyhedron we can find a good approximation of the maximum (minimum) perimeter of a convex polyhedron in an orthogonal projection.

Chapter 5

Optimal visibility ratio for line segments in 2D

In this chapter, we present our algorithms for finding optimal visibility ratio for a set of line segments in 2D. We consider both maximizing the minimum visibility ratio and minimizing the maximum visibility ratio. This problem is relatively easier than our previous three problems. We give $O(n \log n)$ time algorithms to solve the problem. We at first describe the problem formally and then give our algorithms in detail.

5.1 Maximizing the minimum visibility ratio

5.1.1 The problem

Given n line segments in 2D, the problem is to find the direction d in 2D for which the minimum visibility ratio is maximized . The visibility ratio r_l for line segment l_i as already defined in Chapter 2 is just $\sin \theta_{(d,l_i)}$ where $\theta_{(d,l_i)}$ is the acute angle between d and l_i. Therefore, our goal is to find the optimal direction d where $d = max_x\{min_{l_i}\{\sin \theta_{(x,l_i)}\}\}$

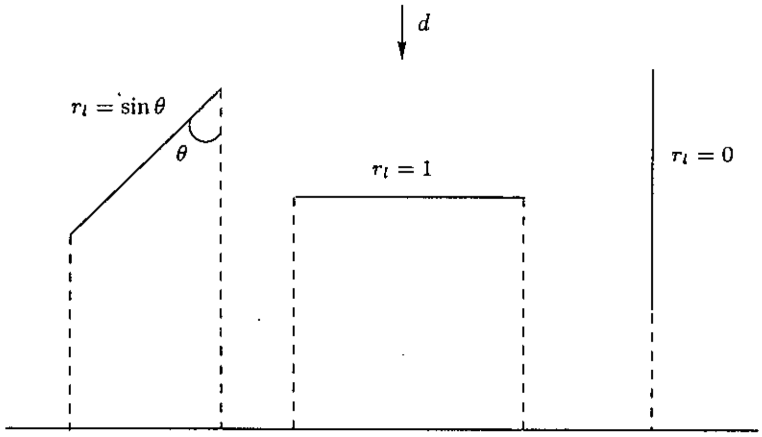


Figure 5.1: Visibility ratio r of line segments.

5.1.2 Mapping the problem to unit circle

An alternate setting of this problem is some what easier to understand and so we adopt this strategy of mapping the line segments into an origin centered unit circle. Since visibility ratio does not depend upon the length of the line segment we just work with $\theta_{(d,l_i)}$. Let, h_i's be the lines that pass through the origin and parallel to l_i's. Let, C be an unit circle centered at the origin. All the h_i's divide C into a number of disjoint conical regions C_H. Two such cones are separated by exactly one line. Any point x on the perimeter of C corresponds to a direction vector from the center of C to x. Henceforth, we use x to denote this vector. Now the visibility ratio for any line segment l_i is $\sin\theta_{(x,h_i)}$, where $\theta_{(x,h_i)}$ is the acute angle between x and h_i. Since $\theta_{(x,h_i)}$ is acute and $\sin\theta \propto \theta$ for $\theta \leq 90^0$, we only consider to maximize the minimum $\theta_{(x,h_i)}$. Therefore, our goal is to find the optimal direction d where $d = max_x\{min_{h_i}\{\theta_{(x,h_i)}\}\}$.

5.1.3 Finding the optimal direction

Let us at first find out the optimal direction for a particular cone C_H. Each C_H is bounded by exactly two lines. Let us denote a cone that is bounded by h_i and h_j as $C_{H(i,j)}$. Now we state the following:

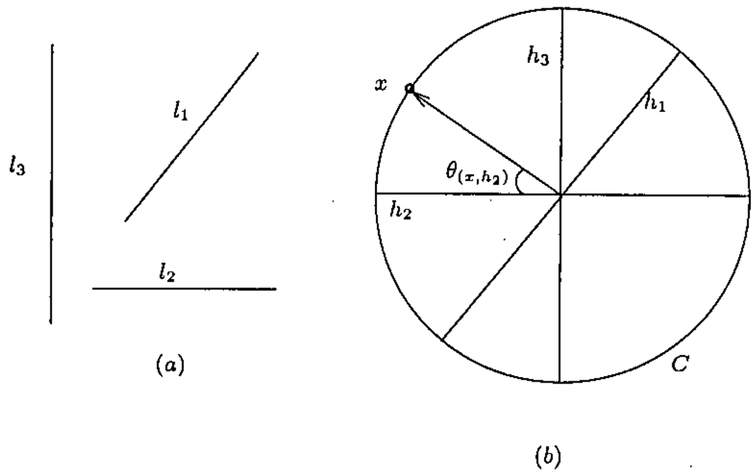


Figure 5.2: (a) Original line segments, and (b) line segments after mapping to circle are shown.

Lemma 5.1.1. *For a given cone $C_{H(i,j)}$, the direction that maximizes the minimum visibility ratio is the bisector of the angle formed by h_i and h_j.*

Proof. By the definition of $C_{H(i,j)}$, all other lines except h_i and h_j makes larger angle with any direction within the cone. So, the minimum visibility ratio is determined by h_i or h_j only. Now for any direction inside $C_{H(i,j)}$ other than the bisector, either h_i or h_j makes smaller angle and therefore producing a smaller minimum visibility ratio. □

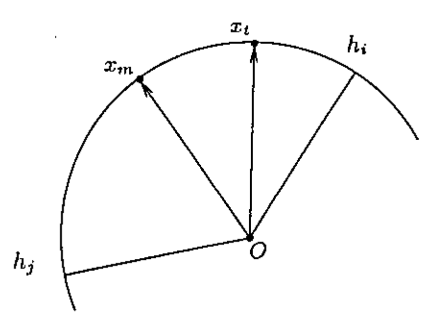


Figure 5.3: ox_m is the bisector of $\angle h_i o h_j$. Any other direction ox_t makes smaller angle with either h_i or h_j.

Lemma 5.1.1 immediately suggests an algorithm that checks the mid point of all the arcs of C_H's to find out the direction that maximizes the minimum visibility ratio. Yet instead of checking all the C_H's we can just take the cone that have the largest arc.

Theorem 5.1.1. *Let d be the bisector of the angle between the bounding lines l_i and l_j of the cone $C_{H(i,j)}$ having largest arc. The minimum visibility ratio is maximized in the direction d and it can be found in $O(n \log n)$ time.*

Proof. The mid point of the arc of a cone denotes the direction for which the minimum visibility ratio is maximized inside that cone and value of the ratio is proportional to the half-angle between the bounding lines. So the global maximum value must be found at the cone whose bounding lines makes the largest angle at the origin. Now, formation of the cones takes $O(n \log n)$ time as we need to sort the lines according to their polar angle. Finding C_H having the largest arc takes $O(n)$ time since there are $2n$ cones in total and the mid-point is obtained in $O(1)$. Thus, the overall time complexity is $O(n \log n)$. □

5.2 Minimizing the maximum visibility ratio

5.2.1 The problem

Given n line segments in 2D, the problem is to find the direction d in 2D for which the maximum visibility ratio is minimized. The visibility ratio for line segment l_i as already defined in Chapter 2 is just $\sin \theta_{(d,l_i)}$ where $\theta_{(d,l_i)}$ is the acute angle between d and l_i. Therefore, our goal is to find the optimal direction d where $d = min_x\{max_{l_i}\{\sin \theta_{(x,l_i)}\}\}$

5.2.2 Mapping the problem to unit circle

Like the maximization case, we again map the problem to unit circle. Let h_i's be the lines that pass through the origin and parallel to l_i's. Let C be an unit circle centered at the origin O. Each h_i intersects C at exactly two points and we get $2n$ intersection points $\{p_1, p_2, \ldots p_{2n}\}$. Let us assume that $\{p_i\}$ are sorted according to their polar angles and we define the set S_k as $\{p_k, p_{k+1}, p_{k+2}, \ldots, p_{k+n-1}\}$ where $1 \leq k \leq n$. The set S_k contains exactly n points and let us define the arc bounded by p_k and p_{k+n-1} as A_k. All the A_k's are disjoint.

Any point x on the perimeter of C corresponds to a direction vector from the center of C to x. Henceforth, we use x to denote this vector. Now the visibility ratio for any line segment l_i is $\sin\theta_{(x,h_i)}$, where $\theta_{(x,h_i)}$ is the acute angle between x and h_i. Since $\theta_{(x,h_i)}$ is acute and $\sin\theta \propto \theta$ for $\theta \leq 90^0$, we only consider $\theta_{(x,h_i)}$.

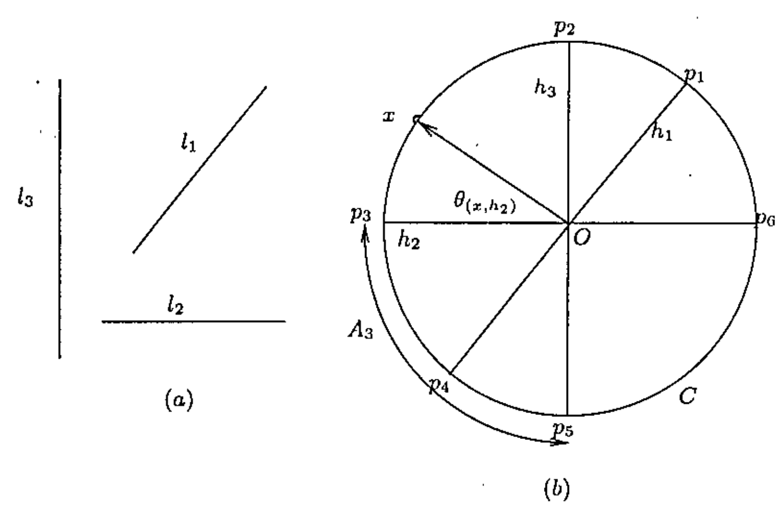


Figure 5.4: (a) Original line segments, and (b) the arc A_3 corresponding to $S_3 = \{p_3, p_4, p_5\}$ are shown.

5.2.3 Finding the optimal direction

We, at first, find the optimal direction for each of A_k and then take the minimum over all $1 \leq k \leq n$. For a particular A_k we state the following:

Lemma 5.2.1. *Within A_k the two end points p_k and p_{k+n-1} determine the direction for maximum visibility ratio.*

Proof. Suppose d_k is the direction for maximum visibility ratio within A_k. Now it cannot make greater angle with any direction other than p_k or p_{k+n-1}. The argument follows. Suppose, point p_j where $k < j < (k+n-1)$ makes the largest angle with d_k. Now p_j lies between p_k and p_{k+n-1}. If both p_j and p_k lie on the same side of d_k then clearly $\angle d_k o p_k > \angle d_k o p_j$. Otherwise both p_j and p_{k+n-1} lie on the same side of d_k and $\angle d_k o p_{k+n-1} > \angle d_k o p_j$. This contradicts to the assumption that p_j makes the largest angle with d_k. Therefore, d_k can make greatest angle with the two end points only. □

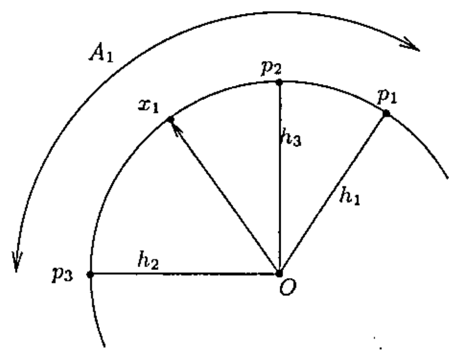


Figure 5.5: A_1 corresponding to $S_1 = \{p_1, p_2, p_3\}$ is shown. ox_1 is the bisector of $\angle p_1 o p_3$ and hence x_1 denotes the optimal direction for A_1.

Lemma 5.2.2. *The bisector d_k of the directions denoted by p_k and p_{k+n-1} is the direction that minimizes the maximum visibility ratio within A_k.*

Proof. From lemma 5.2.1 we know that either p_k or p_{k+n-1} is responsible for the maximum visibility ratio within A_k. Now any direction other than d_k makes greater angle with either p_k or p_{k+n-1} and hence leaving an opportunity to minimize the angle further by moving it towards d_k. At d_k the maximum visibility ratio is minimized. □

From the above lemmas we state the following:

Theorem 5.2.1. *The direction d that minimizes maximum visibility ratio can be found in $O(n \log n)$ time.*

Proof. Finding the $2n$ intersection points take $O(n)$ time. Points are then sorted according to the polar angles. This step is $O(n \log n)$. The maximum visibility ratio within each A_k can be computed in $O(1)$ and we have n such sets; so it will take $O(n)$ time to find the minimum. Hence, the overall running time is $O(n \log n) + O(n) = O(n \log n)$. □

Chapter 6

Conclusion

In this thesis, we studied computing nice projections of some basic objects like line segments and triangles in 2D and 3D for several criteria of niceness. For a set of line segments we have given exact algorithms for finding the optimal projection of line segments in 2D and also given approximation algorithms for line segments in 3D. We have also worked with triangles in 3D and given exact algorithms for finding the optimal projection.

For line segments in 2D, our algorithms run in $O(n \log n)$. We followed McKenna and Seidel's approach closely for this problem by extending the concept of view from convex polyhedra to line segments in 2D. We mapped the problem to unit circle and formed view cones. For each view cone we defined a line with certain property and then showed that all these lines form a centrally symmetric convex polygon. We proved that the direction denoted by the radius of the largest inscribed circle maximizes the sum of projection and the direction denoted by the radius of the smallest circumscribed circle minimizes the sum of projection.

For triangles in 3D, our approach is similar to the one with line segments in 2D. In this problem, we defined similar concept of view for planes in 3D. Each plane parallel to the plane of a triangle and passing through the origin divides the 3D space and forms view cones. For each view cone we defined a plane with certain property and then showed that all these plane form a centrally symmetric convex polyhedron. We proved that the direction denoted by the radius of the largest inscribed sphere maximizes the sum of projected area and the direction denoted by the radius of the smallest circumscribed sphere minimizes the sum of projected area. The running time of our algorithm in $O(n^2)$.

For line segments in 3D, our algorithm is heuristic based. Experimental results show that for the maximization case, the heuristic is a good one and for minimization case it is surprisingly close to brute force optimal solution. The heuristic for minimization case might be a good starting point for further research on finding an exact solution. The algorithms run in $O(n^2)$. Using efficient data structure and some precomputation it might be possible to improve this time complexity.

For the problem of maximizing (minimizing) the minimum (maximum) visibility ratio of line segments in 2D, we have given exact algorithms. Both algorithms run in $O(n \log n)$ time. To solve the problem, we translate all the line segments to the origin. These line segments divide the perimeter of an origin centered circle into a number of arcs. We showed that the direction that maximizes the minimum visibility ratio is denoted by the direction of the mid point of the largest arc. For minimizing the maximum visibility ratio, we take the $2n$ intersection points of the lines with the circle. After sorting these points, we consider every two point at a gap of n within the sorted list and the optimal direction is denoted by the direction of the mid point of those pair of points that are closest.

6.1 Future works

The running time of the problem of finding the maximum (minimum) sum of projection for line segments in 2D is $O(n \log n)$. This running time might be improved to $O(n)$ since there exists an $O(n)$ time algorithm for finding the maximum (minimum) projection of a polygon. So, this problem is worth investigating in future.

For the problem of finding the maximum (minimum) sum of projection of line segments in 3D we have given heuristic based approximate solutions.

For a particular view, our algorithm often outputs a direction that might fall outside the view cone and thus making it necessary to bound it within the cone. Even during considering the line segments incrementally, the intermediate directions may go outside the view cone. So the heuristic can further be modified so that the direction is always within the view cone.In this work, our algorithm is analytical. We cannot guarantee that our solution reaches any local (or global) maxima. To obtain this, genetic algorithm can be applied to it.

In order to generate the brute force optimal solution for our third problem, we used global searching. To generate a better quality solution, an adaptive version of this can be used. After getting the brute force solution as now, we can search for a better one using coarser searching.

In future, this problem can further be studied to find an exact solution. For the minimization problem, the heuristic we have given can be a good starting point to an exact algorithm. An exact solution to these problems will lead to the solution of the problem of finding maximum and minimum perimeter projection of a convex polyhedron, for which we do not know any result. The problem of finding the maximum (minimum) perimeter projection of a convex polyhedron is a special case of the problem of maximum (minimum) sum of projection of line segments in 3D, where the line segments corresponds to the edges of the polyhedron. Considering each of the edges present in each view as a separate line segment in 3D and finding the maximum (minimum) sum of projections of those line segments, we can find the maximum and minimum perimeter projection of a convex polyhedron.

Bibliography

[1] *Computing Nice Projections of Convex Polyhedra*, volume 4921, 2008.

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

[3] T. Biedl, M. Hasan, and A. Lopez-Ortiz. Efficient view point selection for silhouettes of convex polyhedra. In *Proceedings of the 29th International Symposium Mathematical Foundations of Computer Science*, volume 3153 of *Lecture Notes in Computer Science*, pages 735–747. Springer, 2004.

[4] P. Bose, F. Gomez, P. Ramos, and G. Toussaint. Drawing nice projections of objects in space. *Journal of Visual Communication and Image Representation*, 10(2):155–172, 1999.

[5] A. Bottino, L. Jaulin, and A. Laurentini. Reconstructing 3D objects from silhouettes with unknown view points: The case of planar orthographic views. In *Proceedings of the 8th Iberoamerican Congress on Pattern Recognition*, volume 2905 of *Lecture Notes in Computer Science*, pages 153–162. Springer, 2003.

[6] A. Bottino and A. Laurentini. Introducing a new problem: Shape from silhouette when the relative position of view point is unknown. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(11):1484–1493, 2003.

[7] T. Burger and P. Gritzmann. Finding optimal shadows of polytopes. *Discrete & Computational Geometry*, 24(2-3):219–239, 2000.

[8] P. Eades, M. E. Houle, and R. Webber. Finding the best viewpoints for three dimensional graph drawings. In *Proceedings of the 5th International Symposium on Graph Drawing*, volume 1353 of *Lecture Notes in Computer Science*, pages 87–98. Springer, 1998.

[9] H. Edelsbrunner, J. O'Rourke, and R. Seidel. Constructing arrangements of lines and hyperplanes with applications. In *Proceedings of the 24th Foundations of Computer Science Conference*, pages 83–91, November 1983.

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

[11] F. Gomez, F. Hurtado, J. Selleres, and G. Toussaint. Perspective projections and removal of degeneracies. In *Proceedings of the 10th Canadian Conference on Computational Geometry*, pages 100–101, Montral, Qubec, Canada, August 1998.

[12] F. Gomez, S. Ramaswami, and G. Toussaint. On removing non-degeneracy assumptions in computational geometry. In *Proceedings of the 3rd Italian Conference on Algorithms and Complexity*, volume 1203 of *Lecture Notes in Computer Science*, pages 86–99, 1997.

[13] 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.

[14] M. McKenna and R. Seidel. Finding the optimal shadows of a convex polytope. In *Proceedings of the 1st Annual ACM Symposium of Computational Geometry*, pages 24–28, 1985.

Index

Acute angle, 65
Arrangement, 41
Azimuth, 20

Brute force, 51

Cone, 19
    View cone, 19
    Conical region, 19
    View cone, 19, 20
Cuboid, 12

Edge, 17

Face, 17

Geodesic distance, 21

Hemisphere, 20
    Negative hemisphere, 20
    Positive hemisphere, 20
Heuristic, 43

Incremental approach, 46

Latitude, 20
Line segments in 3D, 43

Maximum (minimum) area projection, 12
Minimum crossing projection, 12

Nice projections, 11
    Biedl et al's criteria, 13
    Criteria of niceness, 11
    Bose et al's criteria, 12
    Burger and Gritzman's criteria, 12
    McKenna and Seidel's criteria, 12
    Visibility ratio criteria, 13
Normal vector, 18
    Negative normal, 19
    Positive normal, 19

Plane, 19
Polygon, 17
Polyhedron
    Convex polyhedron, 17
Polytope, 18
    Shadow area of polytope, 23
Projection, 11
    Center of projection, 11
    Orthogonal projection, 11
    Perspective projection, 11
    Planar projection, 17
    Plane of projection, 11

Scene, 11
Silhouette, 13
Spherical coordinates, 20
Spherical polygon, 49

Vertex, 17
View point, 11

Visibility ratio, 13, 22, 61
Maximum visibility ratio is minimized, 64
Minimum visibility ratio is maximized, 61

Glossary

A_k: The arc bounded by p_k and p_{k+n-1}.

B_F: The intersection of cone C_F and plane π_F.

C_F: A view cone corresponding to a set of visible faces.

C_H: A conical region formed when the π_i's divide S.

$C_{H(i,j)}$: A cone that is bounded by h_i and h_j.

F: The set of visible faces.

H_F: The halfspace defined by $\{x \mid x \cdot N_F \leq 1\}$.

K_F: The intersection of C_F and H_F.

N_F: The sum of normals within a view cone.

N_f: An outward normal of a facet.

P: A convex polyhedron.

R: Radius of S_p.

S: An origin centered unit sphere.

S_k: A set of points $\{p_k, p_{k+1}, p_{k+2}, \ldots, p_{k+n-1}\}$ where $1 \leq k \leq n$.

S_p: A smallest circumscribed sphere (or circle) of Y_p.

S_{CH}: The spherical portion of C_H that is bounded by the π_i's.

Y_p: A convex polytope or polyhedron.

$\hat{d}_n$: The approximate optimal direction after bounding d_n within a view.

π: A plane.

π_F: The plane which is perpendicular to N_F but displaced at a distance of $\frac{1}{|N_F|}$ away from the origin.

π_i: A plane that is orthogonal to l_i and passes through the origin.

$\pi_{(l_i,l_j)}$: The plane that is parallel to the plane containing l_i and l_j and passes through the origin.

θ: An angle between the view direction and the outward normal; an acute angle between the line segment l and the direction d.

$\theta_{(d,l_i)}$: The acute angle between d and l_i.

$\theta_{(i,x)}$: The acute angles that l_i makes with ox.

$\theta_{(x,h_i)}$: The acute angle between x and h_i.

c: A great circle.

c_i: A great circle formed by $\pi_i \cap S$.

d: A unit direction vector.

d_k: The direction for maximum visibility ratio within A_k.

d_n: An approximate optimal direction after considering n-th line segment.

d_{max}: A direction that maximizes the sum of projections.

d_{min}: A direction that minimizes the sum of projections.

f: A 2D facet of polyhedron.

h_f: A plane that is parallel to a face and pass through the origin.

h_i: The positive hemisphere of l_i; lines that pass through the origin and parallel to l_i's.

l: A line segment. The i-th line segment is l_i.

n: Number of line segments or triangles.

n_i: The visible normal of plane π_i.

p: A point.

p_k: The k-th point of a sorted list of points.

r: Radius of s_p.

r_l: Visibility ratio of a line segment.

s_p: A largest inscribed sphere (or circle) of Y_p.

t: A triangle.

v: A single vertex.

x: A point on the origin centered unit sphere (or circle) denoting a direction from the origin to x.