Reconstruction and Visualization of Polyhedra Using Projections

by

Masud Hasan

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2005

Abstract

Two types of problems are studied in this thesis: reconstruction and visualization of polygons and polyhedra.

Three problems are considered in reconstruction of polygons and polyhedra, given a set of projection characteristics. The first problem is to reconstruct a closed convex polygon (polyhedron) given the number of visible edges (faces) from each of a set of directions $\mathcal{S}$. The main results for this problem include the necessary and sufficient conditions for the existence of a polygon that realizes the projections. This characterization gives an algorithm to construct a feasible polygon when it exists. The other main result is an algorithm to find the maximum and minimum size of a feasible polygon for the given set $\mathcal{S}$. Some special cases for non-convex polygons and for perspective projections are also studied.

For reconstruction of polyhedra, it is shown that when the projection directions are co-planar, a feasible polyhedron (i.e. a polyhedron satisfying the projection properties) can be constructed from a feasible polygon and vice versa. When the directions are covered by two planes, if the number of visible faces from each of the directions is at least four, then an algorithm is presented to decide the existence of a feasible polyhedron and to construct one, when it exists. When the directions see arbitrary number of faces, the same algorithm works, except for a particular sub-case.

A polyhedron is, in general, called equiprojective, if from any direction the size of the projection or the projection boundary is fixed, where the "size" means the number of vertices, edge, or faces. A special problem on reconstruction of polyhedra is to find all equiprojective polyhedra. For the case when the size is the number of vertices in the projection boundary, main results include the characterization of all equiprojective polyhedra and an algorithm to recognize them, and finding the minimum equiprojective polyhedra. Other measures of equiprojectivity are also studied.

Finally, the problem of efficient visualization of polyhedra under given constraints is considered. A user might wish to find a projection that highlights certain properties of a polyhedron. In particular, the problem considered is given a set of vertices, edges, and/or faces of a convex polyhedron, how to determine all projections of the polyhedron such that the elements of the given set are on the projection boundary. The results include efficient algorithms for both perspective and orthogonal projections, and improved adaptive algorithm when only edges are given and they form disjoint paths. A related problem of finding all projections where the given edges, faces, and/or vertices are not on the projection boundary is also studied.

Acknowledgements

I wish to thank my two excellent supervisors, Alejandro López-Ortiz and Therese Biedl, for guiding my research throughout my PhD studies. Their encouragement, friendship, and financial support are invaluable for me. Most of the results in this thesis also come from the joint work with them.

I am grateful to Anna Lubiw, who first introduced me to the problems related to polyhedra. Some results in this thesis come from the joint work with her.

I like to thank my other examiners, David Clausi, Jeff Erickson, and Craig S. Kaplan, for their valuable comments, suggestions and corrections.

I also like to thank Jonathan Buss for his guidance in the early days of my PhD studies.

Finally, I would like to express my gratitude to the School of Computer Science for providing such a beautiful environment.

Contents

List of Figures

Chapter 1

Introduction

Problems involving three-dimensional objects are often solved by exploiting the relation between an object and its projection, such as in reconstruction of objects from their images, rendering objects using silhouettes, object recognition by matching features in images, or effective visualization of an object by projections representing desired features. In this thesis, among the problems just mentioned, two types of problem have focused: *reconstruction of polyhedra from projections* and *visualization of polyhedra via projections.*

Reconstructing polyhedra from projection information is an important field of research due to its applications in geometric modeling, CAD, computer vision, object recognition, geometric tomography, and computer graphics. The nature of different reconstruction problems and the techniques to solve them depend upon the types of information given, which include line drawings, images, silhouettes, area/volume and shapes of shadows, texture, shading [35, 80], and albedo[1] [65]. Based on the amount of information given it may not be possible to uniquely

[1]The proportion of light or radiation reflected by a surface.

reconstruct a polyhedron, since more than one polyhedron may have similar projections. In that case all possible polyhedra are reconstructed from the geometric information alone, and, if necessary, the best approximation is picked up with the help of some other secondary information [80].

The problem of constructing new polyhedra based on certain mathematical properties have also been extensively studied since antiquity in the fields of architecture, art, ornament, nature, cartography, and even in philosophy and literature. (See the book [25] and the web page [40] for interesting discussions on the history of discovering new polyhedra.) In this regard, regularity of faces, edges and vertices, and symmetry are some properties that have been extensively studied. For example in a platonic solid all faces are the same regular convex polygon and all vertices are incident to the same number of faces.

Projections are an efficient way of conveying succient three-dimensional information of an object, which is in turn useful in object visualization, object recognition, and automatic visual inspection of objects. For example, projections are used in representing three-dimensional graph drawings and visualizing knots. In object recognition different features of an object are obtained from different projections and then matched with stored features. In automatic visual inspection, it is important to find projections which highlight features to be verified.

We study two types of problems or reconstruction of polyhedra, deciding the existence of feasible polyhedra that realizes some given projections and construing new polyhedra given some projection properties. The projection information that we consider are the quantitative measures of different features in the projections, such as the number of edges and faces in the whole projection as well as in the projection boundary.

For the problems on visualization we consider algorithms for efficiently finding projections of polyhedra. We impose required properties on the projection boundary, particularly those related to the inclusion of a given set of edges, vertices and faces on the projection boundary.

1.1 Basic definitions

A *polygon* in two dimensions (2D) is the region bounded by a simple circuit formed by finite number of line segments called *edges*. A polygon is *convex* if a line segment connecting any of its two points is entirely inside of it, otherwise it is *non-convex*. A *polyhedron*, which is the 3D counterpart of a polygon, is the region bounded by a finite number of polygons called *faces* such that (i) if two faces intersect, then it is only at a common edge or a vertex, (ii) every edge of every face is an edge of exactly one other face, and (iii) faces surrounding each vertex form a simple circuit [23]. A polyhedron is *convex* if a line segment connecting any of its two points is entirely inside of it, otherwise it is *non-convex*. In a different way, a convex polygon is the bounded region defined by the intersection of a finite number of half-planes and a convex polyhedron is the bounded region defined by the intersection of a finite number of half-spaces [87].

A *projection* of a polyhedron P from a view point p (from a *view direction* d) is the *mapping* of every visible point of P onto the projection plane, where the "mapping" of a point is the intersection of the line passing through that point and p (the line passing through that point and is parallel to d) with the projection plane. The projection of a polygon P has the same definition with "projection plane" replaced by "projection line". If a projection is defined with respect to a view point, then it is called *perspective* and if it is defined with respect to a view direction, then it is called *orthogonal.* See Figure 1.1 for an example. The view direction of an orthogonal projection points towards the origin (instead of pointing away from the origin.)

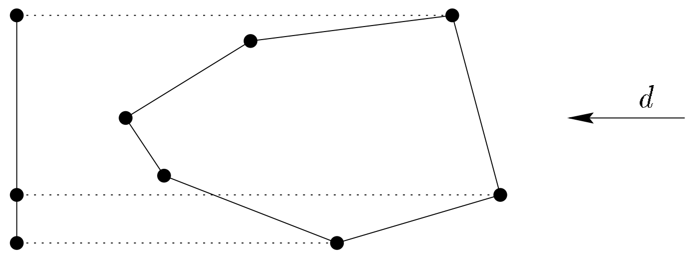


Figure 1.1: An orthogonal projection of a convex polygon.

In a perspective projection of a convex polygon P, an edge e of P is *visible* from the view point p if and only if P and p are in two different half-planes of the line containing e. In an orthogonal projection of P, e is *visible* from the view direction d if and only if the inner product of d with the outer normal of e is negative. Similarly, for a perspective projection of a convex polyhedron P, a face f is *visible* from the view point p if and only if P and p are in two different half-spaces of the plane of f. For an orthogonal projection, f is visible from the view direction d if and only if the inner product of d with the outer normal of f is negative.

The similar definitions for non-convex polygons (polyhedra) are less formal, since the non-convex cases are not studied in depth. For a non-convex polygon, an edge e is *visible* (*invisible*) if and only if every point of e is visible (invisible), and e is *partially visible* if some but not all points of e are visible. Similarly for a non-convex polyhedron, a face f is *visible* (*invisible*) if and only if every point of f is visible (invisible), and f is *partially visible* if some but not all points of f are visible.

A projection of a convex polygon is a line segment, possibly divided into smaller line segments that are mappings of visible edges. Similarly, for a convex polyhedron a projection is a convex polygon, possibly divided into smaller convex polygons that are mappings of visible faces. The *projection boundary* of a convex polyhedron P is the set of edges in the projection each of whose corresponding edge in P has exactly one face visible and the other incident face invisible.

The concept of *silhouette* is very close to that of projection boundary and can be defined in general for both convex and non-convex polyhedra. In a projection of a polyhedron P, the silhouette is the set of visible pieces of edges each of whose corresponding edge in P has one adjacent face invisible and the other adjacent face visible or partially visible. Observe that for a convex polyhedron, the silhouette is the same as projection boundary and we will use these two terms as synonyms.

When a view point or a view direction changes, the set of visible faces in the projection of a convex polyhedron P may or may not change. Over all projections of P, each distinct set of visible faces is called a *view* of P. The set of all view points or view directions for which the projections have the same view is called a *viewing region*. Note that a viewing region is always a connected set and for orthogonal projections it can be considered as a cone with its apex at the origin.

A convex polygon or polyhedron, or more generally a convex object in 2D or 3D, is *centrally symmetric* if it is symmetric to itself with respect to a point called the *center of symmetry* [53, Page 37]. For example cube is centrally symmetric, whereas a pyramid is not. A *parallel-sided* $2m$*-gon* is a convex polygon with $2m$ edges such that the edges are in parallel pairs. Observe that a parallel-sided $2m$-gon is centrally symmetric if the edges in each parallel pair are of equal length. A *zonohedron*, or more formally a *generalized zonohedron*, is a convex polyhedron whose faces are centrally symmetric [81]. See Chapter 2 for more on zonohedra.

1.2 Problems considered in this thesis

Problems on reconstruction of polyhedra

As mentioned briefly at the beginning, for the reconstruction of polyhedra the projection information that has been considered previously is usually the exact geometry of the projections such as triangulations, line drawings, and silhouettes and different geometric measures of the projections such as volume of the projections along with some non-geometric surface information such as shading, texture, and albedo. (See more on related works in Chapter 2.) In contrast, a very different type of projection information, which is also very limited, is considered in this thesis. The information for reconstructing polygons is only a given number of visible edges in some projections, and for reconstructing polyhedra this is only a given number of visible faces in some projections. The main focus is on convex polygons and convex polyhedra and on orthogonal projections.

Although from the application point of view the problem of reconstructing polyhedra is more common than that of reconstructing polygons, surprisingly, the problems on reconstructing polygons that are considered in this thesis are themselves very rich and their results and solution techniques will serve as foundation for solving the analogous problems that are considered for reconstructing polyhedra. Given the number of visible edges in some projections, the necessary and sufficient conditions for the existence of a feasible polygon are given first. The proof of sufficiency also gives an algorithm to construct a feasible polygon. Then, based on this characterization, an algorithm to find the maximum and minimum size of a feasible polygon is presented. Finally, the construction of non-convex polygons and the case of perspective projections are studid in brief.

With the 2D results in hand, we will move proceed to the problem of reconstruction of polyhedra given the number of faces in the projections. When all the view directions lie in a single plane in 3D, we will show that the characterization and reconstruction algorithms for feasible polygons can be used in the feasible polyhedra case. When the directions are in more than one plane, the problem bocomes more complicated, since the interaction among the neighbouring directions increases. Only the case of two planes is considered in this thesis. For this case an efficient algorithm for reconstructing feasible polyhedra is presented, which works for all but one sub-case.

The avobementioned problems on reconstruction are theoretically and intrinsically interesting. On the other hand, from an application point of view it may be advantageous for several reasons to consider only the number of edges/faces in the projections (which we consider for our problems). Quite often projections are created by human beings, as opposed to extracting them from pictures, in the form of line drawings [80, 84]. In that case the number of visible edges or visible faces in the projections can be provided without drawing the actual projections and thus simplifying the reconstruction process. Else, if the information is extracted from pictures, then still extracting only the number of edges/faces should be easier and more accurate compared to extracting other information like the line drawings, since the user does not need to worry about the exact geometry of the projection. As a further advantage, storing and handling only some integers should be much simpler and faster.

Apart from the aforementioned advantages, our characterization of feasible polyhedra can be useful as a preliminary step in applications in which other type of information is used for reconstruction purposes— the user can decide quickly the existence of possible resulting polyhedra before starting a complicated and time consuming reconstruction process.

As a final remark in favor of the problems considered here, to our knowledge there is no work where only the number of visible edges/faces in projections have been considered for reconstruction puposes.

Reconstruction of a special type of polyhedra

Consider the problem of reconstruction of polyhedra when every projection has the same information. More precisely, how to re construct a polyhedron when the number of vertices, edges and/or faces in every projection is the same? The feasible polyhedra for this case are the so-called equiprojective polyhedra. The main problem here is to reconstruct all equiprojective polyhedra. Although this problem may be considered as a special type of the earlier problems of reconstruction (mentioned in the previous subsection), part of it was separately known in mathematics since 1968 as an open problem. Shephard in his paper “Twenty problems on convex polyhedra” [72] first posed the problem of constructing all *boundary-vertex equiprojective polyhedra.* A k-boundary vertex equiprojective polyhedron is one whose number of vertices in all projection boundaries is k. Later, Croft, Falconer, and Guy mentioned this problem in their book “Unsolved Problems in Geometry” [24].

As a first attempt to the problem posed by Shephard, we give a characterization and recognition algorithm for boundary-vertex equiprojective polyhedra [41]. Then we find minimal boundary-vertex equiprojective polyhedra, where the minimality is in terms of the number of vertices in the projection boundary.

The pioneering definition of boundary-vertex equiprojective polyhedra given by Shepherd [72] considers only the number of vertices in the projection boundary. Inspired by the initial results on boundary-vertex equiprojective polyhedra, we extend the definition of equiprojective polyhedra from boundary-vertex to *visible-vertex*, *visible-edge*, and *visible-face equiprojective polyhedra*[2]. Under these new measures of equiprojectivity, a polyhedron is k-visible vertex (similarly k-visible edge or k-visible face) equiprojective if the number of visible vertices (similarly number of visible edges or number of visible faces), is k for any projection. Our results on these new types of equiprojective polyhedra include the relation among different types and discovering that generalized zonohedra fall into these three classes.

Visualization of polyhedra

For visualization of polyhedra, the following problem is considered: given a convex polyhedron and given some property of the silhouette, how hard is it to find one or all projections that have the property? This question is motivated by numerous applications of silhouettes, such as rendering in computer graphics and object recognition in computer vision (see Chapter 2 for more on silhouette applications). Two applications specifically benefit from the ability to bring certain features such as edges or faces on the silhouette. In quality control of a manufacturing process such as casting, checking for flaws such as air pockets can be done by examining whether each edge is a smooth and continuous line. This can be done efficiently if edges appear on the silhouette, using video cameras to acquire the silhouette of the part. In visualization, crucial features should be forced to the silhouette to make them easily detectable. Also, if features are to be labeled it is advantageous to move them to the silhouette, since the outside area allows for space to place labels.

A straightforward approach to attack the aforementioned problem is to compute all possible views and then check each view for desired properties, but that would be inefficient. We give generalized algorithms to find all projections of a convex

[2]J. O'Rourke first mentioned the idea of this extension of equiprojective polyhedra in 15 th Canadian Conference on Computational Geometry (CCCG), August, 2003, Halifax, Nova Scotia.

polyhedron such that a given set of edges, faces and/or vertices appear and/or do not appear on the silhouette. For orthogonal projections and for edges only, the algorithm is fully adaptive in the number of disjoint paths and thus improve the time complexity.

1.3 Organization of this thesis

In Chapter 2 gives some background. Reconstruction of polygons and polyhedra are in Chapters 3 and 4, respectively. Chapter 5 deals with equiprojective polyhedra. Visualization of polyhedra is in Chapter 6. Finally, Chapter 7 concludes this thesis with open problems and future work.

Chapter 2

Background

This chapter discusses in more details the related works for the following topics: reconstruction of polyhedra, views of polyhedra, silhouette computation, and efficient visualization of 3D objects. Some background on zonohedra is also presented.

2.1 Reconstruction of polyhedra

There has been substantial work in reconstruction of polyhedra from projections. In particular, the computational geometry community has studied certain theoretical aspects of this problem, much of it motivated by Steinitz' characterization of the *edge-graph* of a convex polyhedron. In the edge-graph of a convex polyhedron P, there is exactly one vertex for each vertex of P and two vertices are connected by an edge if and only if the corresponding vertices in P form an edge. Steinitz' characterization states that a graph is the edge-graph of a convex polyhedron if and only if it is simple, planar, and 3-connected. The "if" part of his proof is constructive, which means that given a simple 3-connected planar graph, it is always possible to construct a convex polyhedron whose edge-graph is the given graph. See [87, Chapter 3] for a proof, among many that have been published, of Steinitz' theorem. These proofs of Steinitz' theorem can be implemented in $O(n^3)$ time, as noted by Das and Goodrich [26]. They also describe a linear-time algorithm to reconstruct a polyhedron from a given 3-connected and triangulated planar graph.

The following problem was studied by several authors. Given a convex polygon of n vertices and its two distinct triangulations (share no diagonal), construct a convex polyhedron of n vertices such that the two projections from the z axis are the two triangulations and the given polygon is the projection boundary. (See Figure 2.1, where two triangulations realize to a tetrahedron.) Here all the vertices of the polyhedron are in the projection boundary. The technique for constructing the polyhedron is to perturb the vertices in z axis. Guibas conjectured that such a construction is always possible, but this was later disproved by Dekster [28]. Marlin and Toussaint [57] gave an $O(n^2)$ algorithm for deciding whether such a polyhedron exists and constructing a polyhedron where possible.

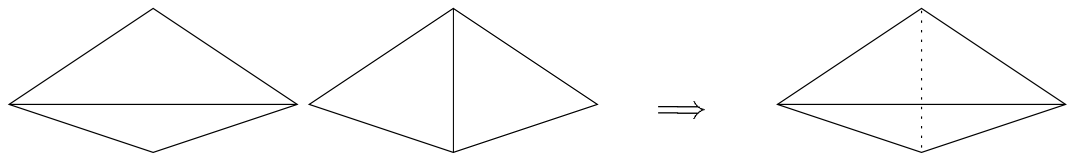


Figure 2.1: Reconstruction of a polyhedron from two triangulations.

In this problem the union of two triangulations corresponds to the edge-graph of a polyhedron. But since the vertex-coordinates are given in xy-plane, the two triangulations may not realize to a convex polyhedron even if the corresponding graph of their union is 3-connected planar. Marlin and Toussaint [57] proved that if it is allowed to change the positions of the vertices of the polygon, then the two triangulations always realize a convex polyhedron. In another variation of this problem, where the triangulations are isomorphic to two opposite projections from z-axis, Bereg [10] showed that the polyhedron can always be reconstructed. See [29] for a collection of similar problems on reconstruction of polyhedra.

Reconstructing polyhedra have got more attention from application point of view. Based on different applications, various projection information have been considered for reconstruction. Among them the line drawings [55, 56, 63, 65, 79, 80, 84, 86] are possibly the mostly considered one. Line drawings may be obtained from images, may be geometric drawings from the designers [80, Chapter 1], or even may be freehand drawings [52, 83]. The reconstruction algorithms differ on whether the reconstruction is from a single drawing or from multiple drawings.

For multiple drawings there are two common approaches based on the representation of polyhedra that are to be reconstructed: constructive solid geometry and boundary representation. Both approaches are used in engineering and product design such as designing complex mechanical parts and CAD [43, 84]. In constructive solid geometry approach each object is represented by a set of primitive objects like cube, prism and pyramid, a set of transformation like translation, rotation and scaling, and a set of boolean operation like union, intersection and difference. The reconstruction process can be represented by a tree, with leaves as primitive objects and internal nodes as boolean operation with associated transformation. The primitive objects are identified, usually by users, from the line drawings [84].

In boundary representation approach an object is represented by its boundary which in turn is represented by a finite number of faces, edges and vertices. The major steps for reconstruction are to create 3D vertices and edges from those in the drawings, then creating faces, and finally finding the adjacency among the faces to create the real object [84].

Although it is comparatively more difficult to reconstruct a polyhedron from a single drawing [80, 84] there is a substantial amount of work on it. Here the reconstruction process has two major steps. The first one is to decide whether a given line drawing correctly represents a polyhedron. For example the line drawing in Figure 2.2(a) does not represent any polyhedron since the dotted extensions of four edges do not meet a point, although it looks like a truncated square pyramid. During this verification step all possible 3D information are also collected. The next step is the construction of the polyhedron from those information.

There are several approaches for deciding the validity of a line drawing. Labeling the line drawing is one such approach. Huffman [44] and Clowes [22] independently presented a pioneering labeling scheme where the idea is to find a correct labeling so that the convex, reflex, and silhouette edges are labeled with "+", "-", and "$\rightarrow$", respectively. See Figure 2.2(b) for an example. This scheme was later extended by several authors to include hidden lines and curves in the line drawings [79, 80, 84].

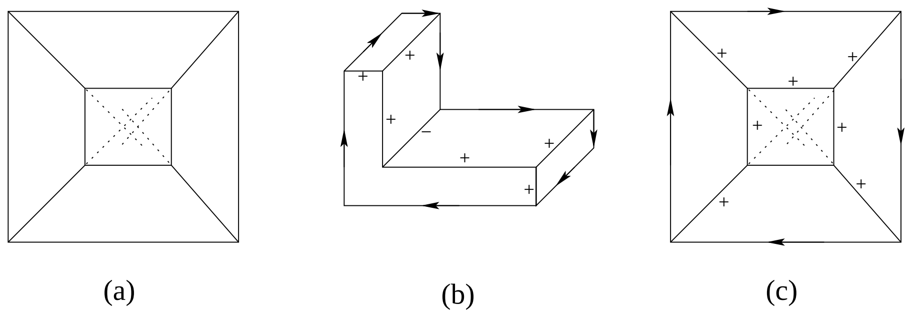


Figure 2.2: (a) An incorrect line drawing of a truncated tetrahedra. (b) Example of a valid labeled line drawing. (c) A valid labeling of the incorrect line drawing in (a).

For a line drawing to represent a polyhedron, it is necessary to have a consistent labeling, but it is not sufficient [79, 80, 84]. For example Figure 2.2(c) shows a correct labeling of the incorrect line drawing of Figure 2.2(a). Several ideas have been proposed for the sufficiency of a line drawing [44, 54, 80, 84]. Sugihara have given necessary and sufficient conditions in term of linear programming for a labeled line drawing to represent a polyhedron [79, 80]. Another idea is the use of reciprocal figure in gradient/dual space where the faces, edges and vertices correspond to vertices, edges and faces in the original line drawing [44, 54, 80, 84]. According to this idea a incorrect line drawing can not have a reciprocal figure.

Among other projection information that have been considered for reconstruction of polyhedra the area and shape of projections have been considered in geometric tomography [35]. Usually the convex objects are reconstructed here. Geometric tomography deal with more on the mathematics of reconstruction rather than its application. A related but more application oriented field is computerized tomography, where 3D objects are reconstructed from sectioning information such as area of a plane section, of the objects. Medical CAT scanner is an important application of computerized tomography where image of a part of a human body can be reconstructed from X-rays [35]. Although projections and X-rays are not the same, the information achieved through X-rays can give the lengths, widths, volumes and shapes of different parts of an object, which are similar to area and shape of projections.

Instead of whole projections, sometimes only silhouettes are used to reconstruct polyhedra [13, 51, 58]. For example, in silhouette probing, which is one kind of geometric probing, the only information available is a set of silhouettes that are discovered interactively by probing the object. See the PhD thesis [75] and two related articles [76, 77] by Skiena for a detailed discussion on reconstruction by geometric probing. In volume intersection, which is a well-known technique in computer vision, the only information available for constructing a polyhedron is a set of silhouettes [12, 13, 51], sometimes even with unknown view points [12, 13].

Problems of reconstruction of polyhedra have two different flavors based on whether the input is achieved interactively or not. In interactive reconstruction, the object is "proved" by a mathematical or physical measuring device (such as measuring lengths) and the position of a subsequent prove may depend upon the measurements achieved from the previous probes. The problem is to reconstruct the object using fewest number of probes. In non-interactive reconstruction, all the inputs are known beforehand. See also [76, 77] for more on both approaches.

Discovering and studying new polyhedra based on certain mathematical properties still continues to this date, while it started with platonic solids at forth century BC [25, Chapter 2]. For example, there are algorithms to construct zonohedra [33, 34, 40]. In another example, Kaplan and Hart [45] introduced an infinite class of convex polyhedra called *symmetrohedra*. The polyhedra in this class are constructed by placing regular polygons at the rotational axes of a polyhedral symmetry group and filling the "gap" by placing polygons of a non-regular type. This class contains a substantial number of new polyhedra.

2.2 Views of polyhedra

The number of all possible view points and view directions from which a polyhedron can be projected are infinite. But the number of views of a polyhedron for both perspective and orthogonal projections is finite.

For a convex polyhedron with n faces, this number is $\theta(n^2)$ for orthogonal projections and $\theta(n^3)$ for perspective projections [36, 37, 66, 70]. These bounds are in fact for the viewing regions, each of which represents a view. Since a view changes when one or more faces change their visibility, the viewing regions are defined by the planes of the faces. Each cell in the the arrangement of the planes of the faces gives a viewing region, and it can be proved that this arrangement has $\theta(n^3)$ cells in total with $\theta(n^2)$ cells unbounded. The unbounded cells, when translated to the origin, consists of view directions and represent views for orthogonal projections.

For a convex polyhedron, computing the viewing regions is straight forward and takes $\theta(n^2)$ and $\theta(n^3)$ time for perspective and orthogonal projections, respectively, [36, 37, 66, 70]— for perspective projections simply compute the arrangement of the planes of the faces of the polyhedron optimally in $\theta(n^3)$ time [68] and for orthogonal projections translate the planes to the origin and then compute the arrangement in $\theta(n^2)$ time [68].

For a non-convex polyhedron, the definition of view is lot more complicated. (Since the views of non-convex polyhedra are not used in this thesis, its definition is omitted. Only some ideas on how the views and their bounds are computed are given.) There are several *events* in the projection that can change a view, such as a face becomes a line, a vertex intersects a line, and three edges from three different faces intersect into a point (see Figure 2.3). Observe that the first among the above three is the only event that can happen for a convex polyhedra. See [36, 37, 66, 70] for other possible events.

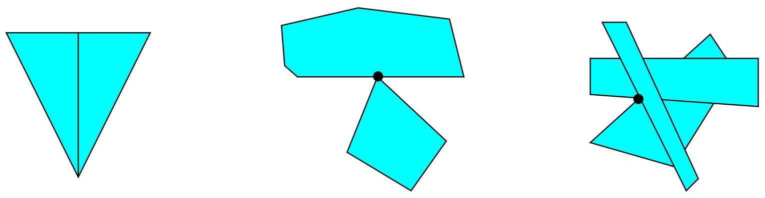


Figure 2.3: Three events that change the view of a non-convex polyhedron.

Computing viewing regions for non-convex polyhedra is also complicated. For a particular event the set of all view points or view directions are determined by the type of the event. For example, if the event is due to a face becoming a line, then the view points and view directions lie in the plane of that face, whereas if the event is due to three edges intersecting into a point, then the view points and view directions lie in a hyperboloid. In general the space of view pointsfor which events occur are quadratic surfaces [36, 37, 66, 70]. The number of such surfaces depends upon the number of triples consisting of edges or vertices or both, and for a polyhedron of size n this number is $\theta(n^3)$. Moreover, the arrangement of those surfaces gives $\theta(n^9)$ cells with $\theta(n^6)$ of them unbounded. The time complexity to compute all viewing regions by computing this arrangement takes $\theta(n^9)\log n$ and $\theta(n^6)\log n$ for perspective and orthogonal projections, respectively [36, 37, 66, 70].

There is a graph-theoretic representation called *aspect graphs* for views of polyhedra [70]. In an aspect graph, a vertex represents a view and two vertices are connected by an edge if and only if the two corresponding views are adjacent, where by "adjacent" means the two viewing regions have common boundary. For example see Figure 2.4, where the view in (a) has three adjacent views shown in (b), and the corresponding subgraph is shown in (c).

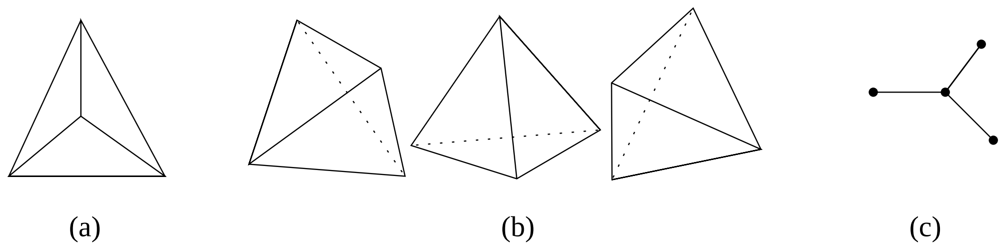


Figure 2.4: Four adjacent views and corresponding subgraph of aspect graph.

In other related works, instead of considering whole projections, only silhouettes have been considered to define views and aspect graphs [32, 50, 70, 71]. The views of polyhedral terrains and polyhedral scenes have also been considered [2, 3, 27]. See also [70] and references therein for advantages and disadvantages, variations, and alternates of views and aspect graphs.

2.3 Silhouette computation

Silhouettes are useful in various settings, especially in the area of machine vision, for example in 3D assembly purposes. In an assembly process performed by a machine, it may be required to gauge the given parts. In order to gauge a part, cameras are used to acquire back-illuminated silhouettes of the part along with the location of the light source. The silhouette is processed so that key features can be identified and the distances among these features can be calculated [61]. Similarly silhouettes are used to compute the boundary and the orientation of the mechanical parts to be picked up by a robot for assembly [60]. Silhouettes are also used for quality control [73, 74], object recognition [73], illuminating critical features [62], and others.

In computer graphics, silhouette edges represent discontinuities in the visibility of an object, and are one of the strongest visual cues of the shape of the object [49]. When rendering the object, it often suffices to render only the silhouette edges, which can result in substantial speedup [67]. This is because for a polyhedron the number of silhouette edges is usually much smaller than the total number of edges [48].

The use of silhouettes in reconstruction processes has already been discussed earlier. In the applications of silhouette, as mentioned above, it is require to compute the silhouette efficiently. The computation of silhouettes has been studied extensively both in the computational geometry community and in the computer graphics community. Pop et al. [67] gave an algorithm for perspective projections that maintains the silhouette of polyhedra efficiently during arbitrary changes of the view point. They define a silhouette edge in terms of its dual (see also Section 6.2.1) to develop a practical and efficient heuristic to maintain the silhouettes.

Efrat et al. [32] presented several combinatorial bounds on the silhouette structure of a collection of convex polyhedra when the view point moves along a straight line or along an algebraic curve. They compute the silhouette map, which is the arrangement of the silhouettes of all objects with their hidden parts removed. Their combinatorial complexity is the bound on the number of combinatorial changes in the silhouette map during the motion of the view point.

For orthogonal projections only, Benichoe and Elber [9] give output-sensitive algorithms to find silhouettes from polyhedral scenes for a given view point. By mapping all view directions onto the surface of a sphere and then mapping the sphere onto the surface of a cube, they reduce this problem to a segment intersection problem in 2D. Using known techniques for solving segment intersection problems from [1, 20], they find the silhouette in time linear in the size of the output.

2.4 Efficient visualization of 3D objects

There are some studies on how to compute “nice” projections of polygonal objects in 3D [11, 38, 82] and 3D graph drawings [30], where criteria for “nice” include minimizing crossings among the edges, monotonicity of polygonal chains, and minimizing coincidences among the edges and vertices. For convex polyhedra the maximum area of projection [59] and for higher-dimensional polytopes the maximum and minimum volume of projection [15, 16] have also been considered as the desired criteria of the projections. Brunet et al. considered the case of computing the view points from which the projections of a given polygonal chain is a convex polygon [14]. They apply this special case to compute efficiently the occlusion properties in a 3D scene.

2.5 Zonohedra

Zonohedra have been studied a lot due to its beautiful mathematical properties. There is some confusion in the definition of zonohedra. The term "zonohedra" was first defined by the Russian crystallographer Fedorov but was later evolved by Coxeter [23] to mean more special cases (see [81] for the history). Coxeter addressed two special cases: all faces are parallelograms and all edges are of equal length. Coxeter called these two types as zonohedra and equilateral zonohedra [23]. A parallelepiped and a cube are two examples of these two types respectively.

A theorem by Alexandrov on centrally symmetric objects state that: a convex polyhedron whose faces are centrally symmetric is itself centrally symmetric [23, Page 28], [53, Page 41]. By this theorem, a zonohedron is centrally symmetric. This also implies that each face of a zonohedron has a parallel pair with corresponding edges parallel and of equal length. For more information on zonohedra, see the web pages [33, 34, 40].

Chapter 3

Reconstruction of Polygons

This chapter considers the two-dimensional case of the reconstruction problem—construct of polygons from the given number of edges in some projections. The main focus is to create convex polygons from orthogonal projections. After defining the problem precisely, the necessary and sufficient conditions for the existence of a feasible polygon of a given size N is presented. The proof of this characterization gives an algorithm to construct a feasible polygon whenever it exists. Based on the necessary and sufficient conditions, the maximum and minimum size of feasible polygons, when the size N is unknown, can be computed.

We also study creating non-convex polygons from orthogonal projections, and convex polygons from perspective projections. For non-convex polygons, if every direction sees at least two edges, then it is always possible to create a feasible polygon. For perspective projections, if the view points are in convex position, then there it is always possibel to create a feasible polygon.

Some results and ideas of this chapter will work as foundations for similar results for the reconstruction of polyhedra in the next chapter (Chapter 4).

This chapter is organized as follows. The problem definition is in Section 3.1, the necessary and sufficient conditions are in Section 3.2, and their proof is in Section 3.3. The maximum and minimum size of feasible polygons is computed in Section 3.4. Finally, Sections 3.5 and 3.6 adress non-convex polygons and perspective projections, respectively.

3.1 Defining the problem

We only consider non-degenerate orthogonal projections, i.e. projections whose view directions are not parallel to the edges of the polygon. A *direction-integer pair*, or simply a *d-i pair*, $\langle d, n \rangle$ consists of a view direction d and a positive integer n. A *d-i set* S is a non-empty set of d-i pairs where no two directions of S are the same or opposite to each other. (This assumption on directions not being opposite to each other is for that we will ultimately generate and then use d-i pairs for all opposite directions too.) A convex polygon P is *feasible* for S if for each d-i pair $\langle d, n \rangle$ in S, d is not parallel to any edge of P and the number of visible edges from d is n. See Figure 3.1, where a d-i set $S = \{\langle d_0, 3 \rangle, \langle d_1, 4 \rangle, \langle d_2, 4 \rangle\}$ is shown in (a) and a corresponding feasible polygon P is shown in (b).

3.1.1 Examples

Before the problem is stated formally, let us see some examples. For a particular d-i set, it may be possible that a feasible polygon only exists for some particular sizes. For example the size of P in the example of Figure 3.1 is eleven and this is the maximum size that a feasible polygon can have for this d-i set S. As we move in a circular order, the angular distance between any two consecutive directions is less

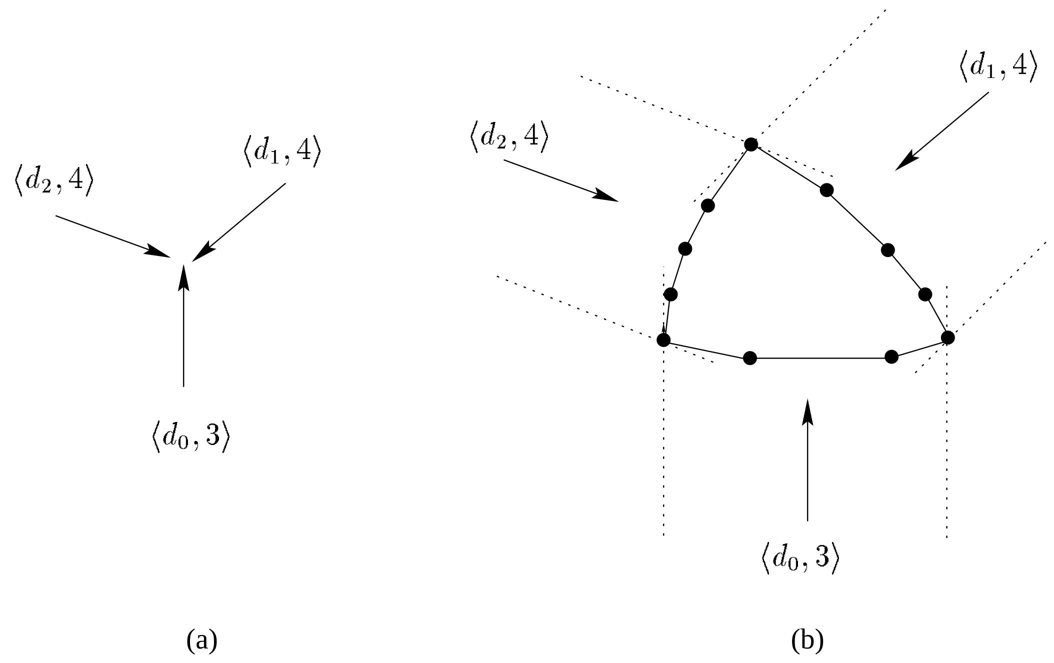


Figure 3.1: (a) A d-i set S. (b) A corresponding feasible polygon$P$ of eleven edges; the pairs of parallel lines that are tangent to P are to distinguish the visible edges for each direction of S.

than π. For any feasible polygon, since the edges are not parallel to the directions, each edge is visible from at least one direction, so the total number of edges that a feasible polygon can have cannot be greater than the sum of the integers of S, which is eleven.

For a particular d-i set, it may also be possible that no feasible polygon exists at all. For example, consider a d-i set S which has the d-i pairs $\langle d_0, 5\rangle$, $\langle d_1, 15\rangle$, and $\langle d_2, 5\rangle$, possibly with some other d-i pairs, such that the direction d_1 is in between the directions d_0 and d_2 in a circular order, and d_0 and d_2 are within $90°$ of d_1 (see Figure 3.2). Then regardless of the polygon size, a feasible polygon for S does not exist. For assume there were such a polygon P. If an edge e is visible from d_1, then the outer normal of e makes an angle less than $90°$ with at least one

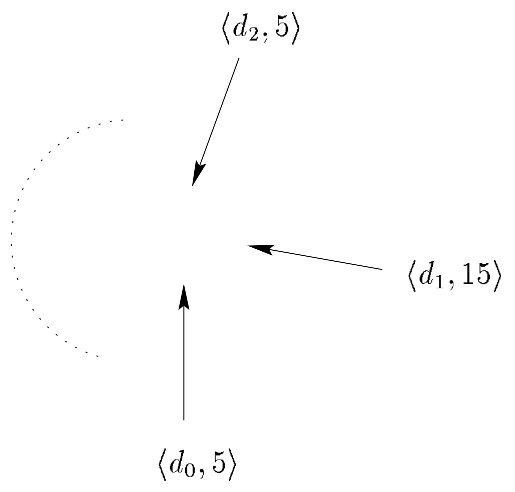


Figure 3.2: A d-i set for which a feasible polygon does not exist.

of d_0 and d_2, and thus e is also visible from at least one of d_0 and d_2. Therefore the total number of edges of P that are visible from d_1 cannot exceed the total number of edges that are visible from d_0 and d_2.

3.1.2 The problem

The above examples illustrate the problems that will be considered for reconstruction of polygons. Let $S = \{\langle d_0, n_0\rangle, \langle d_1, n_1\rangle, \ldots, \langle d_{K-1}, n_{K-1}\rangle\}$ be a d-i set and N be an integer. We ask to create a feasible polygon of size N for S. Clearly, it must be true that $N \geq 3$ and $N > \max_i \{n_i\}$.

For this we first give necessary and sufficient conditions for the existence of a feasible polygon, which also gives an algorithm to construct a feasible polygon when one exists. This algorithm runs in $O(K + N)$ time if S is circularly ordered, either clockwise or counter-clockwise, and $O(K \log K + N)$ time otherwise.

3.2 Necessary and sufficient conditions

We will use, instead of S, a *proper* d-i set $\mathcal{S}$ derived from S. We will prove that a polygon is feasible for S if and only if it is feasible for $\mathcal{S}$. Then we will give the necessary and sufficient conditions for the existence of a feasible polygon for $\mathcal{S}$.

3.2.1 Proper d-i set

Given N, *a proper d-i set* $\mathcal{S}$ is a set of $2K$ d-i pairs, where every d-i pair $\langle d, n\rangle$ has an opposite d-i pair $\langle d', N-n\rangle$ with d' opposite to d. A d-i set S can be turned into a proper d-i set $\mathcal{S}$ in the following natural way. For each d-i pair $\langle d, n\rangle$ in S add both $\langle d, n\rangle$ and $\langle d', N-n\rangle$ in $\mathcal{S}$, where d' is opposite to d. The d-i pairs $\langle d, n\rangle$ and $\langle d', N-n\rangle$ are called *opposite* to each other.

Remember that each direction in S is different and its opposite direction is not in S. So each direction in $\mathcal{S}$ is different and thus the size of $\mathcal{S}$ is $2K$.

A proper d-i set is represented as $\mathcal{S} = \{\langle d_0, n_0\rangle, \langle d_1, n_1\rangle, \ldots, \langle d_{2K-1}, n_{2K-1}\rangle\}$, where the d-i pairs counter-clockwise. Since $0 < n < N$ for each d-i pair $\langle d, n\rangle$ in S, also $0 < n_i < N$ for each d-i pair $\langle d_i, n_i\rangle$ in $\mathcal{S}$. So d_i and d_{i+K} are opposite to each other and and $n_{i+K} = N - n_i$.

Both for the characterization of a feasible polygon and the algorithm for reconstructing a polygon from a given d-i set S, mostly the proper d-i set $\mathcal{S}$ will be used instead of S. From now on the indices of the direction-symbol d and the integer-symbol n of $\mathcal{S}$ are always taken modulo $2K$. The indices of other terms, such as d-arcs, view differences, and final view differences (yet to be introduced) that are derived from $\mathcal{S}$ and their symbols are also taken modulo $2K$.

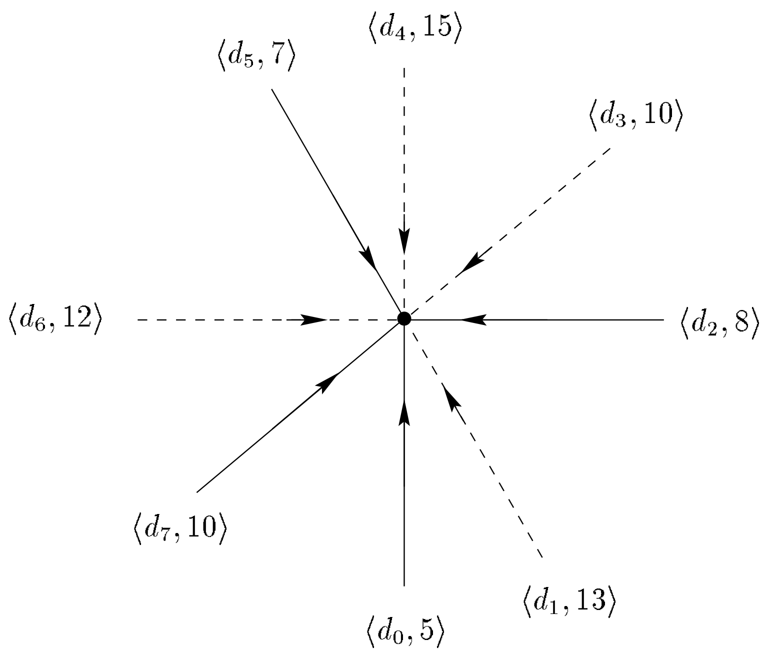


Figure 3.3: A proper d-i set $\mathcal{S}$ created from the given d-i set S given $N = 20$. The d-i pairs whose directions are represented by solid lines form S. All d-i pairs form $\mathcal{S}$.

The *angle* between any two consecutive directions d_i and d_{i+1} in $\mathcal{S}$ is the counterclockwise angle from d_i to d_{i+1} at the origin.

3.2.2 Conditions for S and $\mathcal{S}$ are the same

The feasibility of a polygon for aproper d-i ste $\mathcal{S}$ is defined similarly as it was for S: a convex polygon P is feasible for $\mathcal{S}$ if for each d-i pair $\langle d_i, n_i\rangle$ in $\mathcal{S}$, d_i is not parallel to the edges of P and the number of edges of P that are visible from d_i is n_i.

Lemma 3.1 *A convex polygon P is feasible for S if and only if it feasible for $\mathcal{S}$.*

Proof. ($\Longleftarrow$) This is trivial since S is a subset of $\mathcal{S}$.

($\Longrightarrow$) In $\mathcal{S}$, for any d-i pair $\langle d_i, n_i \rangle$ there is an opposite d-i pair $\langle d_{i+K}, n_{i+K} \rangle$. Exaactly one of them, say $\langle d_i, n_i \rangle$, is in S. Since P is feasible for S, the number of edges that are visible from d_i is n_i. Recall that we assumed that no edge is parallel to a direction in S (and we did this exactly so that this lemma hols.) Since d_{i+K} and d_i are opposite to each other, none of the edges of P is parallel to d_{i+K} either. So the number of edges of P that are visible from d_{i+K} is $N - n_i = n_{i+K}$. So the viewing criteria are satisfied for each d-i pair in $\mathcal{S}$. $\square$

With the above lemma, from now on by a feasible polygon it menas to be feasible for $\mathcal{S}$ unless otherwise stated.

3.2.3 The idea and the condition

Let P be a feasible polygon of size N. The idea of the characterization of P is as follows. Consider the sets of visible edges of P from the directions of $\mathcal{S}$. When we move from a direction d_i to the next one d_{i+1}, there may or may not be any change in the set of visible edges of P, or there may be some edges of P that become *newly visible* and/or *newly invisible* to d_{i+1}. From n_i and n_{i+1} alone, it can not be said exactly how many edges become newly visible or newly invisible to d_{i+1}. However, it is possible to get a lower bound on these quantities.

Observe that if an edge e becomes newly visible when going from d_i to d_{i+1}, then it becomes newly *in*visible when going from d_{i+K} to d_{i+K+1}, which are opposite to d_i and d_{i+1}, respectively. This implies that although the change in the visibility of each edge happens twice, the total change in the visibility for all edges can be counted by only considering their change from invisible to visible. (This use of opposite directions is the main motivation for us to consider the proper d-i set $\mathcal{S}$ instead of the d-i set S.) Moreover, e is newly visible for exactly one direction of $\mathcal{S}$.

The above observations give some idea why, while moving through all directions of $\mathcal{S}$ according to their circular order, the sum of the lower bounds on the number of newly visible edges over all directions can not be greater than the size of P.

We now state the characterization formally. For each i, define the *i-th view difference* as $\delta_i = \mathbf{max}\{0, n_{i+1} - n_i\}$. There must be at least δ_i edges that become newly visible while moving from d_i to d_{i+1}. Therefore if a polygon exists, then $D = \sum_{i=0}^{2K-1} \delta_i \leq N$. Hence the necessary conditions is $D \leq N$ for the existence of a feasible polygon P. Our main result is that this. is also sufficient.

Theorem 3.1 *Given a proper d-i set $\mathcal{S}$ of $2K$ d-i pairs, and given an integer N, where the integers in $\mathcal{S}$ are less than N, a feasible polygon P with N edges exists if and only if $D \leq N$.*

3.3 Proof

3.3.1 Outline of the proof

The proof starts with the following crucial lemma.

Lemma 3.2 *For any i, $n_i - \sum_{j=i+K}^{i-1} \delta_j = \frac{1}{2}(N - D)$.*

Proof. Observe that $n_{i+1} = n_i + \delta_i - \delta_{i+K}$. For if $n_{i+1} > n_i$, then $n_{i+1+K} = N - n_{i+1} < N - n_i = n_{i+K}$, so $\delta_{i+K} = 0$ and $\delta_i = n_{i+1} - n_i$. If $n_{i+1} \leq n_i$, then $\delta_i = 0$ and $\delta_{i+K} = n_{i+K+1} - n_{i+K} = -n_{i+1} + n_i$. Using this K times,

$$N - n_i = n_{i+K} = n_i + \sum_{j=i}^{i+K-1} \delta_j - \sum_{j=i+K}^{i-1} \delta_j.$$

By subtracting the term $D = \sum_{j=0}^{2K-1} \delta_j$ on both sides and re-arranging,

$$N - D = 2n_i - 2 \sum_{j=i+K}^{i-1} \delta_j = 2(n_i - \sum_{j=i+K}^{i-1} \delta_j),$$

which implies the result. $\square$

Note in particular that therefore $N - D$ is even. The idea of the proof can now be outlined as follows. For each view direction d_i, choose δ_i edges such that they are newly visible for d_{i+1}. The remaining $N - D$ edges are chosen in antipodal pairs, so that one becomes visible exactly when the other becomes invisible. To avoid constructing an unbounded polygon we have to be slightly more careful in how to chose edges; we explain this in the next section.

3.3.2 Placing normal-points

Let c be a circle centered at the origin o. Rather than choosing edges directly, choose points on c instead. For a direction d, the (closed) half-circle of c that is visible from d is called the *visible half-circle* of d. A point x on arc a of c is said to be *strictly within* a if it is not an end-point of a.

Let P be an arbitrary convex polygon. The *normal-point* of an edge e of P is the point of c in which the outward normal vector of e, translated to the origin, intersects c. Clearly an edge e is visible from a direction d_i if and only if its normal-point is strictly within the visible half-circle of d_i.

Observe that that the arrangement of the normal-points of P on c implies whether P is bounded or unbounded. If all normal-points intersect a single (open) half-circle of c, the direction for which this is the visible half-circle does not see an edge of P and so P is unbounded. On the other hand, P is bounded means any direction sees at least one edge of P and thus any closed half-circle of c contains at least one normal-point of P strictly within it.

Let us now explain how the normal-points can be placed for each direction, and it needs some more notations. For all i denote by h_i the visible half-circle of d_i. The arc $\theta_i = h_{i+1} \backslash h_i$ is called the *i-th d-arc* ("d" for difference). Normal-points will never be placed on the boundary of θ_i, and hence we will not distinguish carefully as to whether θ_i is open or closed. Observe that the i-th and $(i+K)$-th d-arcs are the reflections of each other with respect to the origin, and are called *opposite* to each other. (See Figure 3.4(a)). Since $d_{i+1} \neq d_i$, θ_i is non-empty. Also $\theta_i \subseteq h_{i+1}$ is at most a half-circle of c. Finally, $\bigcup_{j=i-K}^{i-1} \theta_j = \bigcup_{j=i-K}^{i-1} h_{j+1} \backslash h_j = h_i$. (See also Figure 3.4(b)).

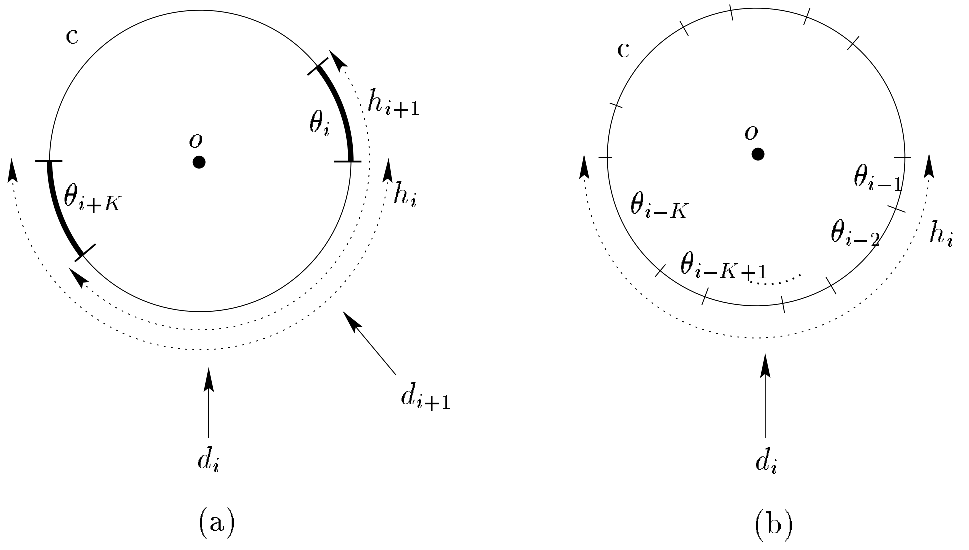


Figure 3.4: (a) Definitions of visible half-circles and d-arcs; two opposite d-arcs are drawn heavily. (b) $h_i = \bigcup_{j=i-K}^{i-1} \theta_j$.

Now place δ_i arbitrary normal-point strictly within each d-arc θ_i. If $D < N$, then by Lemma 3.2 the difference between N and D is even. Select $N - D - 2$

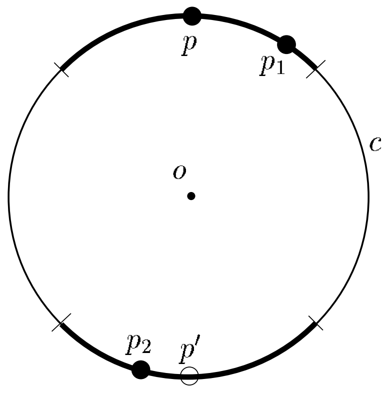


Figure 3.5: Selecting the last two normal points when $D < N$.

additional normal-points in antipodal pairs arbitrarily (but not on end points of any $|theta_i)$ and the remaining two normal-points p_1 and p_2 as follows: Let p be one among $N-2$ already selected normal-points. Let p' be the opposite of p. Select p_1 at clockwise ε circular distance apart from p, and p_2 at clockwise $\varepsilon/2$ distance apart from p'. ε is small enough so that p_1 and p_2 are within two opposite d-arcs. This ends the selection process. See Figure 3.5.

3.3.3 Correctness

To prove that the construction is correct, two things need to be proven: (1) Each direction d_i sees n_i normal-points, i.e., the visible half-circle h_i of d_i gets n_i normal-points strictly within it, and (2) the polygon in bounded, i.e., every open half-circle gets at least one normal-point.

Lemma 3.3 *If $D \leq N$, then each h_i gets n_i normal-points strictly within it.*

Proof. Since each pair among the $N-D$ lastly chosen normal-points goes into two opposite d-arcs, exactly one in that pair is strictly within h_i. Since $h_i = \bigcup_{j=i-K}^{i-1} \theta_j$, the number of normal-points that are strictly within h_i is $\sum_{j=i-K}^{i-1} \delta_j + \frac{1}{2}(N - D)$, which by Lemma 3.2 (with indices modulo $2K$) is n_i . $\square$

Lemma 3.4 *Every open half-circle h of c gets at least one normal-point.*

Proof. If $D < N$, then the last two normal-points p_1 and p_2 were chosen such that the minimum circular distance between any of p, p_1 and p_2 is less than a half-circle, so the lemma holds.

If $N = D$, then each d-arc θ_i gets exactly δ_i normal-points. Among the end-points of the d-arcs, let x_1 and x_2 be the first and last end-points within h in counter-clockwise order. They are not necessarily the end-points of h. (See Figure 3.6). If x_1 and x_2 are the end-points of h, then $h = h_i$ for some i and from Lemma 3.3 h gets $n_i > 0$ normal-points. If x_1 and x_2 are not the end-points of h, then let, for some i, θ_i and θ_{i+K} be the d-arcs that are partially intersected by h and have x_1 and x_2 as one of their end points respectively. Then $\theta_{i+1}, \theta_{i+2}, \ldots, \theta_{i+K-1}$ are strictly within h. Recall from the definition of view difference that either δ_i or δ_{i+K} is 0, say $\delta_i = 0$. So the number of normal-point in h is at least $\sum_{j=i+1}^{i+K-1} \delta_i = \sum_{j=i}^{i+K-1} \delta_i = n_{i+K} > 0$, where the last equality holds by Lemma 3.2 and $D = N$. $\square$

3.3.4 Algorithm

We now summarize the algorithm for constructing a feasible polygon and give its time complexity.

Theorem 3.2 *Given a d-i set S of K d-i pairs ordered counter-clockwise sequence and given an integer $N \geq 3$, where each integer in S is less than N, a feasible polygon P with N edges can be computed, whenever it exists, in $O(N + K)$ time.*

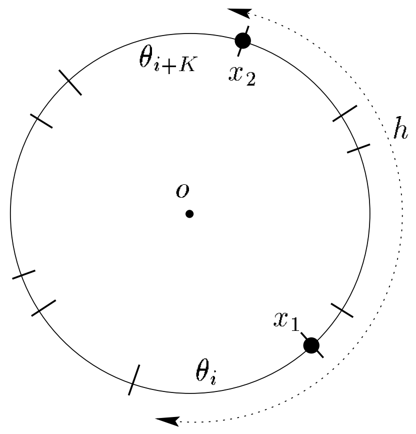


Figure 3.6: Illustrating the proof of Lemma 3.4.

Proof. Compute $\mathcal{S}$ from S. For all i compute δ_i, and from there compute D. If $D > N$, P cannot exist by Theorem 3.1. If $D \leq N$, then select N normal-points from c as described in Section 3.3.2. Finally, compute P by taking tangents through all normal-points and then taking the intersection of all half-planes defined by those tangents and containing c.

Computing $\mathcal{S}$ and its ordering can be done in $O(K)$ time as follows: the ordering of $\mathcal{S}$ is given; from there create another ordered list containing the opposite d-i pairs of S in $O(K)$ time; and finally, merge S with the second list in $O(K)$ time. There are $2K$ d-i pairs in $\mathcal{S}$, so computing D also takes a total $O(K)$ time. Selecting the normal-points on c takes a total of $O(N)$ time. Within the same time, one can keep track of their ordering within each d-arc. Since the ordering of the normal-points are known and since the intersection of every two consecutive tangents according to this ordering gives a vertex of P, P can be computed in $O(N)$ time. So the overall time is $O(N + K)$. $\square$

If S is not sorted, then we sort it in $O(K \log K)$ time and obtain:

Corollary 3.1 *When the set S is unordered, the above algorithm takes $O(N + K \log K)$ time.*

An immediate related problem is: Given a convex polygon P of size N and a d-i set S of K d-i pairs, how fast can we decide whether P is feasible for S? Observe that this problem has a trivial solution of time $O(m \log m)$, where $m = \mathbf{min}\{N, K\}$—traverse the boundary of P and the directions of S in a circular order and use binary search to keep track of the range of directions (or range of edges, based on the minimum among N and K) that sees an edge (similarly that are visible from a direction). If P is allowed to rotate, then the problem becomes more interesting. All possible rotation of P also need to counted, and the time complexity is likely to increase. An interesting open problem is to find algorithm that would solve the above problems in a more efficient way, possibly by using the necessary and sufficient conditions of a feasible polygon (Theorem 3.1).

3.4 Maximum and minimum size of feasible polygons

This section shows show how to find the maximum and minimum size over all feasible polygons for a given d-i set S but unknown N. The algorithm finds the maximum and the minimum simultaneously.

3.4.1 An example

For a particular d-i set S there may be feasible polygons of various sizes. Figure 3.7 shows two feasible polygons of size eleven and nine, respectively, for the d-i set $S =$

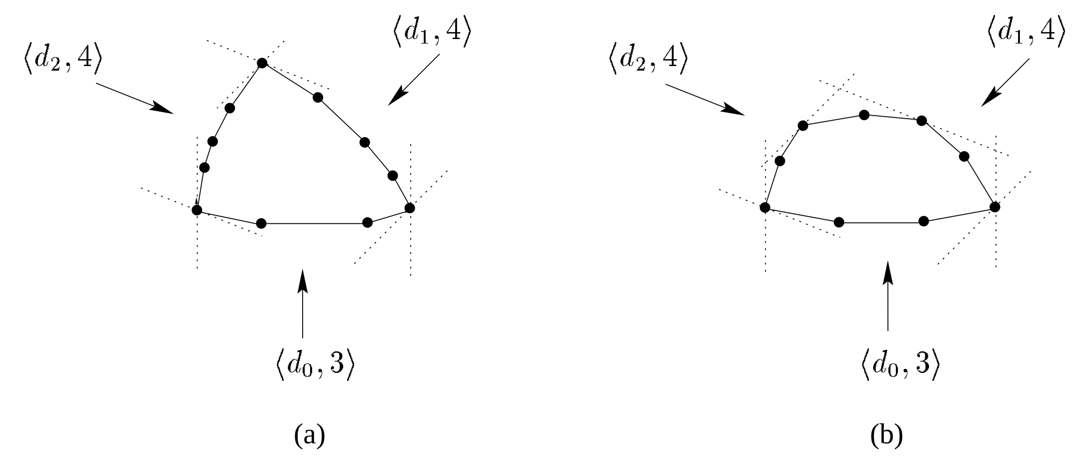


Figure 3.7: Example of feasible polygons with two different sizes for the d-i set $S = \{\langle 0°, 4\rangle, \langle 130°, 4\rangle, \langle 250°, 4\rangle\}$. The sizes are eleven and nine in (a) and (b) respectively.

$\{\langle 0°, 3\rangle, \langle 130°, 4\rangle, \langle 250°, 4\rangle\}$. The reason for different sizes of the feasible polygons is that there may or may not be edges that are commonly visible from more than one direction. For example, for the polygon in Figure 3.7(a) no edge is commonly visible from two or more directions in S, whereas the polygon in Figure 3.7(b) has edges that are commonly visible from two directions in S.

3.4.2 Algorithm outline

As before, N represents the size of a feasible polygon, but N is not given. Observe that if S contains two opposite d-i pairs, then the sum of the two integers would give the value of N. Hence, once again it is assumed that no opposite d-i pair appears in S.

The overall idea of the algorithm is as follows. As an initial step we will compute a proper d-i set $\mathcal{S}(N)$ from S, where the d-i pairs of $\mathcal{S}(N)$ will be functions of N. The view differences δ_i and their sum $D(N)$ also become function of N. Recall from Theorem 3.1 that a feasible polygon exists if and only if $D(N) \leq N$. By analyzing $D(N)$, we find the maximum and minimum values of N such that $D(N) \leq N$.

The algorithm takes $O(K + v \log v)$ time, where K is the number of d-i pairs in S and v is the number d-i pairs in $\mathcal{S}(N)$ that come from S and whose next d-i pairs (in counter-clockwise order), are not from S. Of course $v \in O(K)$, but v could be as small as one if all directions in S are within a half-plane.

3.4.3 Computing the proper d-i set $\mathcal{S}(N)$

We create a proper d-i set $\mathcal{S}(N)$ from S as in Section 3.2.1. Thus for each d-i pair $\langle d, n \rangle$ in S, we let d' be the opposite direction of d and add $\langle d, n \rangle$, and $\langle d', N-n \rangle$ to $\mathcal{S}$, but now N is unknown, and the integer in the second d-i pair is a function depending on N. We call $\langle d, n \rangle$ *original* and $\langle d', N-n \rangle$ *derived.* See Figure 3.8.

As before, $\mathcal{S}(N)$ is represented as $\{\langle d_0, n_0 \rangle, \langle d_1, n_1 \rangle, \ldots, \langle d_{2K-1}, n_{2K-1} \rangle\}$, where the d-i pairs are ordered counter-clockwise by directions. From the above construction, for $0 \leq i \leq 2K-1$, if $\langle d_i, n_i \rangle$ is original (derived), then the opposite d-i pair $\langle d_{i+K}, n_{i+K} \rangle$ is derived (resp. original), and the sum of n_i and n_{i+K} is N.

3.4.4 Expressions for the view differences

Remember how we defined, for $0 \leq i \leq 2K$, the i-th view difference δ_i from two consecutive d-i pairs $\langle d_i, n_i \rangle$ and $\langle d_{i+1}, n_{i+1} \rangle$: $\delta_i = \mathbf{max}\{0, n_{i+1} - n_i\}$. (It is naturally expected that similar to other symbols, the symbol for view differences and the symbol for integers in the d-i pair of $\mathcal{S}$ should be written as function of N, e.g. $\delta_i(N)$, $n_i(N)$. We omit "(N)" is omitted from these symbols for simplicity.)

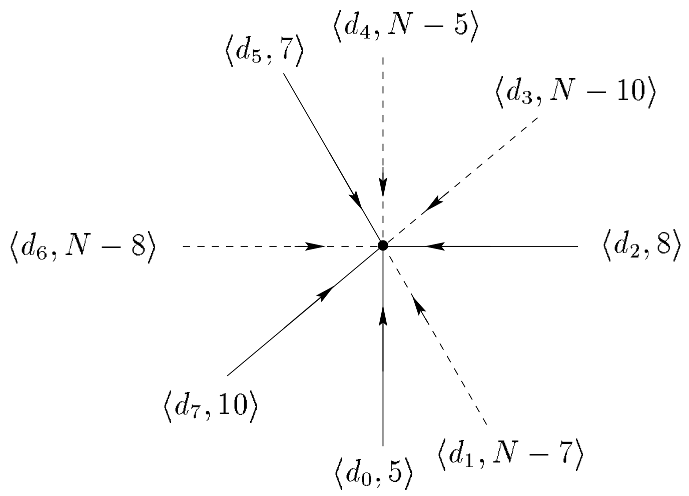

$\langle d_5, 7\rangle$
$\langle d_4, N-5\rangle$
$\langle d_3, N-10\rangle$
$\langle d_6, N-8\rangle$
$\langle d_2, 8\rangle$
$\langle d_7, 10\rangle$
$\langle d_0, 5\rangle$
$\langle d_1, N-7\rangle$

Figure 3.8: The proper d-i set $\mathcal{S}(N)$. The d-i pairs whose directions are represented by solid lines form S.

The following lemma finds the expressions for δ_i.

Lemma 3.5 *For the following four cases, we have*

(i) $\delta_i = \mathbf{max}\{n_{i+1} - n_i, 0\}$, *if both* $\langle d_i, n_i\rangle$ *and* $\langle d_{i+1}, n_{i+1}\rangle$ *are original,*

(ii) $\delta_i = \mathbf{max}\{n_{i+K} - n_{i+K+1}, 0\}$, *if both* $\langle d_i, n_i\rangle$ *and* $\langle d_{i+1}, n_{i+1}\rangle$ *are derived,*

(iii) $\delta_i = \mathbf{max}\{N - (n_{i+K+1} + n_i), 0\}$, *if* $\langle d_i, n_i\rangle$ *is original and* $\langle d_{i+1}, n_{i+1}\rangle$ *is derived, and*

(iv) $\delta_i = \mathbf{max}\{n_{i+1} + n_{i+K} - N, 0\}$, *if* $\langle d_i, n_i\rangle$ *is derived and* $\langle d_{i+1}, n_{i+1}\rangle$ *is original,*

where all the terms except N *are known.*

Proof. Recall that $\delta_i = \mathbf{max}\{0, n_{i+1} - n_i\}$. Replacing n_i by n_{i+K} if $\langle d_i, n_i\rangle$ is derived, and n_{i+1} by n_{i+K+1} if $\langle d_{i+1}, n_{i+1}\rangle$ is derived yeilds the result after basic manipulations. $\Box$

So, $D(N) = \sum_{i+1}^{1K-1} \delta_i$ is a sum of piecewise linear function, and it comes as no surprise that "$D(N) \leq N$?" can be tested easily. We give an especially fast algorithm to do so in the following.

3.4.5 Plotting $D(N)$

Note that δ_i (as function of N) is *increasing* if δ_i falls into case(iii), *decreasing* if it is in case(iv), and *constant* otherwise. Observe that δ_i is increasing if and only if δ_{i+K} is decreasing, and in such cases their two corresponding lines meet at $N = n_i + n_{i+K+1}$, and at which both of them are zero. Such a meeting point is called a *valley* for both δ_i and δ_{i+K}. See Figure 3.9. The total number of valleys is denoted by v. v is what will determine the run time of our algorithm.

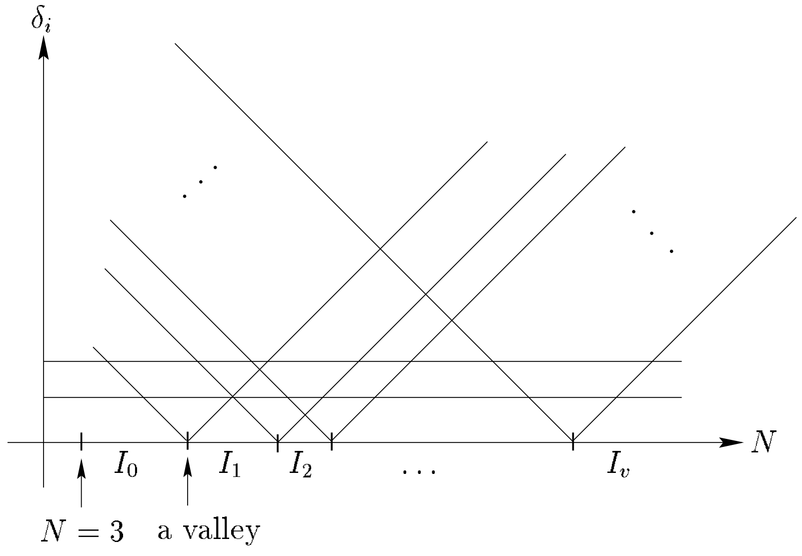


Figure 3.9: Plotting δ_i against N.

Let valleys occur at $N = N_1, N_2, \dots, N_v$, respectively, in increasing sequence. These values divide the real line into $v + 1$ *intervals* with $I_0 = \{N | 3 \leq N \leq N_1\}$,

$I_1 = \{N | N_1 \leq N \leq N_2\}, \ldots, I_v = \{N | N \geq N_v\}$. (See Figure 3.9). Clearly $D(N)$ is linear within each interval; the following lemmas will determine its slope and abscissa.

Lemma 3.6 *In interval I_0, $D(N) = -v \cdot N + C$, where $C = \sum_{i:\delta_i \text{constant}} \delta_i + \sum_{i:\delta_i \text{decreasing}} (n_i + n_{i+K})$.*

Proof. Each constant view difference contributes δ_i to $D(N)$ throughout, and each decreasing view difference δ_i contributes $(n_{i+1} + n_{i+K} - N)$ by Lemma 3.5. Increasing view differences contribute notheing since we are before the first valley. The result follows since there are v decreasing view diferences. Also see Figure 3.10. □

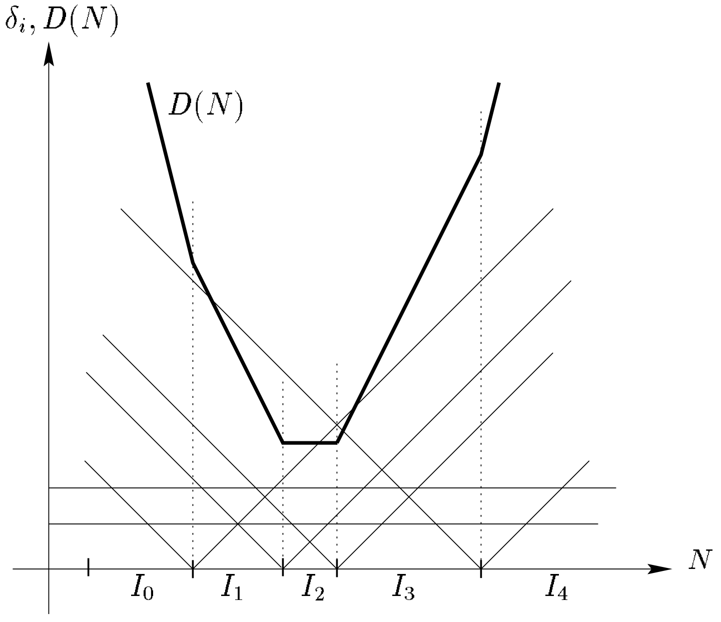


Figure 3.10: Plotting $D(N)$ against N.

Lemma 3.7 *In interval $I_j, j \geq 1$, the slope of $D(N)$ is $D_j(N) = D_{j-1}(N) + 2$, where $D_0(N) = -v$.*

Proof. In I_0, the slope of $D(N)$ is $D_0(N) = -v$ by previous lemma. For $1 \leq j \leq v$, when N moves from I_{j-1} to I_j, one decreasing view difference becomes zero and the corresponding increasing view difference becomes positive. This means the slope of $D(N)$ increases by two from I_{j-1} to I_j, so $D_j(N) = D_{j-1}(N) + 2$. Also see Figure 3.10. $\square$

3.4.6 Finding the maximum and minimum

A *feasible value* of N is the one for which a feasible polygon exists. All feasible values of N are primarily determined by the necessary and sufficient conditions in Theorem 3.1, which is $N \geq D(N)$. The only other criteria for a feasible N is to ensure that N is not too small, namely $N \geq N_{\text{Low}} = \mathbf{\max}_i \{n_i | \langle d_i, n_i \rangle \in S\} + 1$ to ensure that the derived d-i pairs have positive integers.

We are now ready to summarize the algorithm that finds the maximum and minimum N.

Theorem 3.3 *Given a nonempty d-i set S of K d-i pairs ordered counter-clockwise, then the maximum and minimum size N of a feasible polygon can be computed in $O(K + v \log v)$ time, where v is the number of original d-i pairs in $\mathcal{S}(N)$ whose corresponding next d-i pairs are derived.*

Proof. First compute the ordered proper d-i set $\mathcal{S}(N)$ from S in $O(K)$ time as in theorem 3.2. During the creation of $\mathcal{S}$, one can remember which d-i pairs are original and which d-i pairs are derived. Then according to Lemma 3.5 compute the expressions for all $2K$ view differences, and find all valleys in $O(K)$ time. Then sort the valleys to find the intervals of N in $O(v \log v)$ time.

According to Lemmas 3.6 and 3.7 compute C and $D_j(N)$, and from there the expressions of $D(N)$. $D(N)$ is convex, so intersecting it with the line of slope one gives the interval where $D(N) \leq N$. This takes $O(v)$ time. Next, find N_{Low} in $O(K)$ time and add the inequality $N \geq N_{\text{Low}}$, which gives the final interval of feasible values of N. $\square$

If S is not sorted, we sort it in $O(K \log K)$ time and get:

Corollary 3.2 *When the set S is not ordered, the above algorithm takes $O(K \log K)$ time.*

3.5 Non-convex polygons

In this section we briefly study the problem of creating non-convex polygons from orthogonal projections, but first let us see how it differs from that for convex polygons.

3.5.1 Unique features

The issue of creating non-convex polygons from projections differs in many ways from that of creating convex polygons. One such difference is in the visibility of edges. For an edge of a convex polygon, either all points are visible or all points are invisible. For a non-convex polygon, from a view point p the visibility of an edge e which is not collinear with p may be partial due to obstructions created by other edges. See Figure 3.11(a). So when creating a non-convex polygon from projections, how to consider the partially visible edges— are they considered as visible or as invisible?

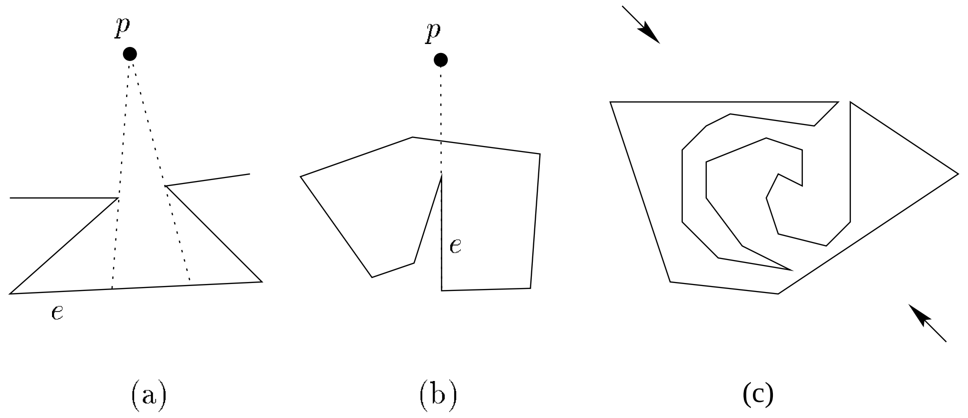


Figure 3.11: (a) The edge e is partially visible from the view point p. (b) The edge e is collinear with the view point p but no end-points of e is visible from p. (c) Two opposite orthogonal projections may not see all edges of a non-convex polygon.

Another difference is in the degeneracies among the view points and the edges of the polygon. For a non-convex polygon, an edge e may be collinear with a view point p while neither of the two end-points of e is visible from p (see Figure 3.11(b)) while for a convex polygon this type of situation cannot happen. So again, when creating a non-convex polygon from projections, how to consider these types of projections — are they considered as degenerate projections or allowed as non-degenerate projections?

Further differences may appear regarding the size of the polygon. For any convex polygon, any two non-degenerate opposite orthogonal projections determine the size of it. But this is not true for non-convex polygon, as there may be many edges that are “hidden” from a d-i pair of opposite directions. (See Figure 3.11(c).)

Keeping the above differences in mind, in this section we will study creating non-convex polygons with some particular assumptions, which include that the size of the feasible polygon is not given, the partially visible edges are not counted as visible, and all directiona see at least two edges. With the above assumptions, we will show that it is always possible to construct a feasible polygon, whose size will be linear in the sum of the number of visible edges from all directions. Reconstructions involving alternative interpretations of visible edges, degeneracies, known size of the feasible polygon, and/or arbitrary visibility are left as open problems.

3.5.2 The construction

The result on non-convex polygons is the following theorem. (Here the model of computation is a real-RAM.)

Theorem 3.4 *Let S be a set of K d-i pairs, where $K \geq 2$. For any d-i pair $\langle d, n \rangle$ in S, assume n, which is the number of visible edges from d, is at least two. Let N be the sum of the integers in S. Then there always exists a non-convex polygon P of size $O(N)$ such that for each d-i pair $\langle d, n \rangle$ in S the edges of P that are not parallel to d and are visible from d is n. Moreover, such a polygon can be constructed in time $O(N)$ if S is ordered according to a circular sequence of its directions.*

Proof. The overall idea of the construction is as follows. First take a very “skinny” parallel-sided hexagon so that no direction in S is parallel to the edges of P. P, being a parallel-sided $2m$-gon in more general term, has the property that any direction sees exactly three edges of P (see Chapter 5 for detail on this argument.) Also ensure that one pair of parallel edges of P, which are called *transient edges*, are very very small in length and for each of them the two internal angles at the two vertices are more than a right angle. (The reason for choosing two edges in that way will be clear later.) At this stage P is called a *base polygon*.

If all integers in S are three, then P is the resulting polygon. Otherwise divide the directions of S into two groups based on which transient edge they see. Then for each group not having all integers three, replace the respective transient edge by a convex polygonal chain as follows. Put, inside of P, an arc which is almost a circle and has the transient edge as its chord. Then for each d-i pair $\langle d, n\rangle$ in that group, find a "visible region" in the arc that is visible only to d and put the necessary number of edges in that region. See Figure 3.12.

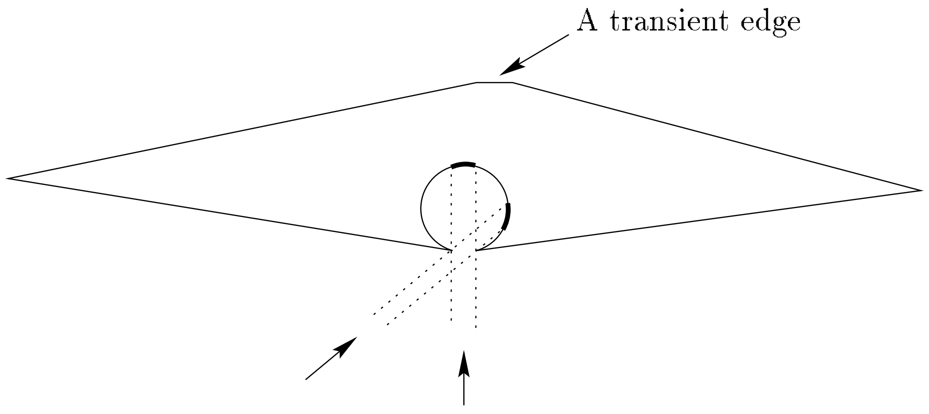


Figure 3.12: The base polygon with one of its transient edges replaced by a circular arc. The unique visible regions for two directions in this arc has been shown in bold.

Let us now examine the construction in detail. Create a base polygon P as decribed above. Let the edges of P be e_1, e_2, $\ldots$, e_6, with e_1 and e_4 being transient.

Assume that not all integers are three. Partition the d-i pairs of S into two subsets: S_1 are d-i pairs with directions that see e_1, and S_2 are d-i pairs with directions that see e_4. Consider the set S_1. If any integer in S_1 is not three, then replace e_1 by a polygonal chain c_1 as follows. First replace e_1 by a circular arc a_1 inside of P so that e_1 is a chord of a_1. Remember that the two internal angles at the two ends of e_1 are more than a right angle, so a_1 is inside of P. The radius of the circle of a_1 is small enough so that if a similar arc a_2 is drawn in place of e_4, then a_1 and a_2 do not intersect.

Now for each d-i pair $\langle d, n\rangle$ of S_1, do the following. Compute the *visible region* of d in a_1. This region can be computed by considering two lines that are parallel to d and pass through the two end-points of e_1, and then taking the arc from a_1 that is "trapped" between these two lines. Not that the length of e_1, which is the size of the "entrance" of a_1, is chosen to be so small that for any two consecutive directions in S_1, their corresponding visible regions in a_1 do not overlap, not even at a common boundary point. Since the number of directions in S_1 is finite, it is always possible to find such a small entrance of a_1 (or equivalently such a small length for e_1). Now, divide this visible region d into $n-1$ pieces and put one vertex somewhere near the middle in each piece of arc. Note that a_1 has other regions in between d-i pairs of consecutive visible regions, call them *invisible*. Put one vertex for each such invisible region.

After putting vertices for all d-i pairs, connect them sequentially by edges to form a chain. Finally, connect the two end-points of (the deleted) e_1 to the two end vertices of this chain by two edges. This ends the construction of c_1 (see Figure 3.13 for an illustration).

Similarly handle the d-i pairs of S_2 and replace e_4 by a polygonal chain c_2 if necessary. This ends the construction of P.

Justification. Consider an arbitrary d-i pair $\langle d, n\rangle$ in S_1. Apart from e_2 and e_6, the edges of c_1 that are visible from d are exactly the edges that have both end-

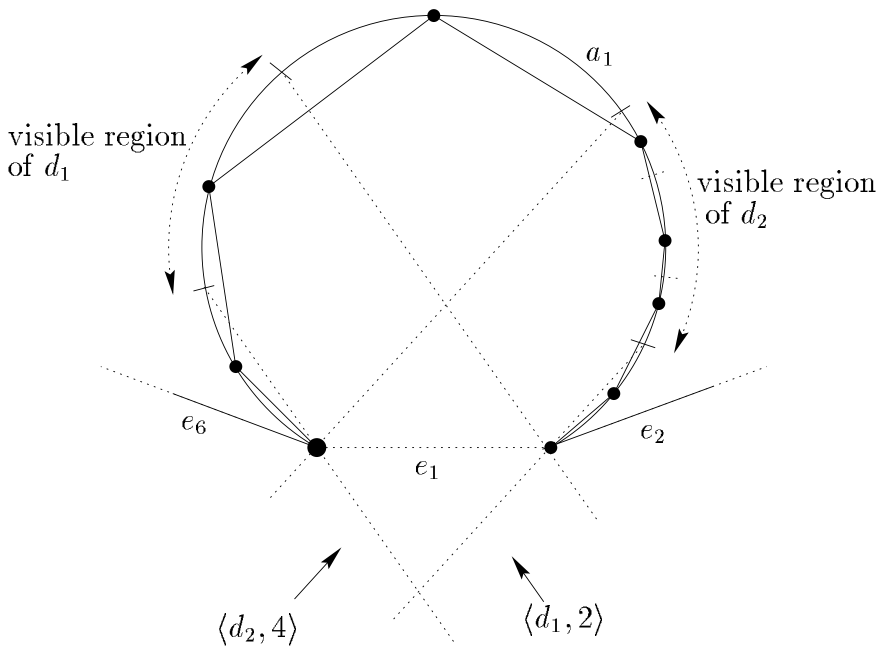


Figure 3.13: Here $s_1 = \{\langle d_1, 2\rangle, \langle d_2, 4\rangle\}$. The edges of P whose vertices are in a_1 are shown. The two edges of the base polygon that are visible to all directions of S_1 are shown as incomplete lines due to limited space.

vertices in the visible region of d. There are $n - 2$ such edges. Note that there are two edges in c_1 for which only one vertex is in the visible region of d. These two edges are partially visible and thus, from the assumption, are not counted as visible from d, The edges of c_2 are invisible from d and from the other four edges of P exactly two which are adjacent to c_1 are visible from d. So over all edges of P the number of edges that are not parallel to d and are visible from d is n. Finally, both c_1 and c_2 are inside of P and do not intersect each other. So P is simple and bounded.

Let us now justify the time complexity and the size of P. Creating P as the initial equiprojective base polygon of size six should take $O(K)$ time. Checking the integers in all d-i pairs takes $O(K)$ time.

For each d-i pair $\langle d, n\rangle$ in S, finding the visible region of d in a_1 or a_2 (as appropriate) takes constant time. For this d-i pair, putting $n-1$ vertices in the visible region takes $O(n)$ time. At the same time one can remember their circular ordering. The number of invisible regions in a_1 and a_2 is proportional to the number of directions in S. So putting all vertices on c_1 takes a total of $O(N+K)$ time, where N is the sum of all integers in S.

From the ordering of the d-i pairs of S, one can find the circular ordering of the visible and invisible regions in each of a_1 and a_2, and the circular ordering of the vertices in each visible region is already known. So one can create c_1 and c_2 by connecting the vertices according to their circular ordering in $O(N+K)$ time. Finally, $K \in O(N)$. So the total time is $O(N)$.

The size of the polygon is linear on the total number of edges in c_1 and c_2 which is in turn linear on the sum of all integers in S. So the size of P is $O(N)$. □

Corollary 3.3 *When the set S is not ordered according to any sequence, the above algorithm takes $O(N + K\log K)$ time.*

Proof. Sort the d-i pairs of S by the directions in $O(K \log K)$ time. □

3.6 Perspective projections

We briefly study the problem of creating convex polygons from perspective projections. We first see how the problem in this case differs from that of the orthogonal case. Then we will give a construction for projections with a particular restriction on the view points.

3.6.1 Comparing with orthogonal case

There are some considerable differences between the problem of creating convex polygons from perspective projections rather than orthogonal projections. For example, the size and "position" of the feasible polygon for orthogonal projections is scaling- and translation-invariant, but for perspective projections it is not. On the other hand, for perspective projections if there are n edges of P that are visible from a view point d, then after scaling up, scaling down, or translation it is highly likely that the number of visible edges of P from d is no longer n. For scaling up the number of visible edges will decrease and for scaling down it will increase. For translation it may change arbitrarily. See Figure 3.14(a) for an example where the number of visible edges of a polygon from a view point is one before scaling and is two after scaling down.

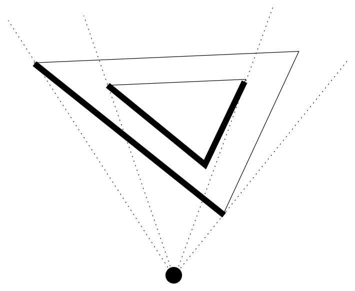


Figure 3.14: A polygon for perspective projection is not scaling-invariant. The outer polygon has been scaled down to the inner polygon. The bold lines are the visible edges from the view point drawn as a circle.

The problem of scaling-dependency gives rise to another problem, which is important from a practical point of view. For perspective projections, the size of a feasible polygon may be too small. But since it is not scaling-invariant, it cannot be scaled up. So for practical purposes the one may not get a reasonable size of the feasible polygon.

3.6.2 View points in convex position

Keeping the difficulties for perspective projections in mind, we study a special case in some detail. In this special case we assume that the given view points are in convex position. This assumption is motivated by a practical point of view, where it may be easy for a camera to take projections of a polygon from the view points in convex position.

For perspective projections a direction-integer d-i pair $\langle d, n\rangle$ in S consists of a *view point* d and a positive integer n. A convex polygon is *feasible* for S when for any d-i pair $\langle d, n\rangle$ in S the number of visible edges from d is n. Like with orthogonal projections for convex polygons (from Sections 3.1 to 3.3.4), only non-degenerate projections are considered. Moreover, similar to the non-convex polygons (Section 3.5), there is no restriction on the size of the feasible polygon.

With the above assumptions, we observe that there always exists a feasible convex polygon of size equal to the sum of all integers in S. In the following section we show how to create such a feasible polygon.

3.6.3 The construction

The main result for perspective projection is the following theorem.

Theorem 3.5 *Given a d-i set S of size $K \geq 3$, where all the view points are in convex position, it is always possible to create a convex polygon P of size N such that for each d-i pair $\langle d, n\rangle$ in S the edges of P are not collinear with d and the* *number of visible edges of P from d is n. Moreover, N is the sum of all integers in S.*

Proof. Let $S = \{\langle d_0, n_0\rangle, \langle d_1, n_1\rangle, \ldots, \langle d_{K-1}, n_{k-1}\rangle\}$, where the ordering of the d-i pairs are according to the counter-clockwise sequence of the view points in their convex hull h. So the vertices of h are $d_0, d_1, \ldots, d_{K-1}$ in counter-clockwise order.

Create a "dual" polygon h' of h as follows. For each edge of h find its middle point. For each vertex d_i in h create an edge e_i in h' by connecting the two middle points of the two adjacent edges of d_i. See Figure 3.15. Note that the size of h' is also K.

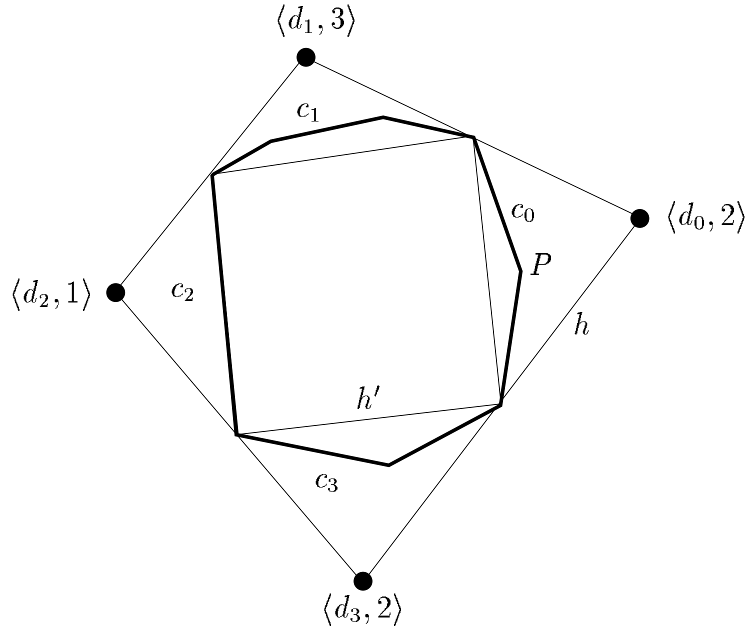


Figure 3.15: Illustrating the proof of Theorem 3.5. Here h is the outermost bigger polygon, h' is the innermost quadrilateral, and P is the polygon drawn in bold lines.

Now convert h' to P. For each $0 \leq i \leq K-1$, replace e_i by a convex chain c_i of size n_i such that the edges of c_i are visible only from d_i and the resulting polygon is still convex. This is possible if the chain c_i is entirely within a connected region of $h - h'$.

The correctness of the above construction lies in the fact that e_i is the only edge in h' which is visible from d_i and that c_i is entirely within $h - h'$. So ultimately d_i sees only the n_i edges of c_i. Moreover, the size of P is $\sum_{i=0}^{K-1} n_i$, which is N. $\square$

Chapter 4

Reconstruction of Polyhedra

This chapter considers the generalization of the previous problems to three dimensions. We study the problem of reconstruction of polyhedra from the given number of faces in some projections. The 2D-results from the previous chapter will work as a foundation for 3D-results of this chapter.

Similar to 2D, the primary focus is on creating convex polyhedra from orthogonal projections. The first case considered is that the view directions are in one plane. In this case, a feasible polyhedron can be constructed from a feasible polygon and vice versa, which will imply that the necessary and sufficient conditions for the existence of a feasible polyhedron, and the algorithm to find maximum and minimum size of a feasible polyhedron also follow from 2D.

Next, the view directions are considered to be covered by two planes. The construction becomes more complicted here. We give an algorithm that always constructs a feasible polyhedron, if it exists, except for one sub-case.

Finally, constructing non-convex polyhedra from orthogonal projections and convex polyhedra from perspective projections are studied in brief. In these two cases the ideas are similar to that in 2D.

This chapter is organized as follows. The reconstruction problem is intoduced in Section 4.1. Section 4.2 gives some preliminaries. Section 4.3 and 4.4 deal with the cases when the directions are in one plane and two planes respectively. Finally, Section 4.5 and 4.6 briefly address the problem for non-convex polyhedron and for perspective projections respectively.

4.1 The reconstruction problem

Given a d-i set S a convex polyhedron P is *feasible* for S if for each d-i pair $\langle d, n\rangle$ in S, d is not parallel to any face of P and the number of visible faces from d is n. For example see Figure 4.1, where S has four d-i pairs $\langle d_0, 4\rangle$, $\langle d_1, 4\rangle$, $\langle d_2, 3\rangle$, and $\langle d_3, 1\rangle$ and the three directions are coplanar. In this example, the feasible polyhedron is a pyramid and the visible faces from d_0 and d_1 are the four triangular faces, from d_2 are two triangular faces and the rectangular face, and from d_3 are the rectangular face only.

It should be conceivable that as in 2D for some particular d-i set S there may not be any feasible polyhedron because of the "mismatch" among the direction and integers in the d-i pairs in S. For example, consider a d-i set S which has the d-i pairs $\langle d_0, 100\rangle$, $\langle d_1, 5\rangle$, $\langle d_2, 5\rangle$, $\langle d_3, 5\rangle$, and $\langle d_4, 5\rangle$, possibly with some other d-i pairs, such that the directions d_1, d_2, d_3, and d_4 surround the direction d_0 very closely. See Figure 4.2 where the directions are represented by points on a sphere s. Then there may not exist any feasible polyhedron for S. It may be possible that for any feasible polyhedron P for S, if a face f of P is visible from d_0, then it is visible from at least one of d_1, d_2, d_3, and d_4. Therefore, the total number of faces

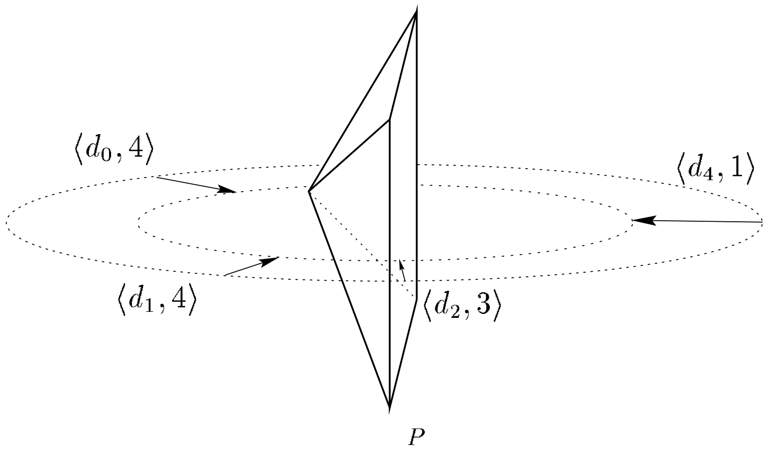


Figure 4.1: Example of S and P where the directions of S are planar.

of P that are visible from d_1, d_2, d_3, and d_4, which is twenty, cannot always be more than the number of faces that are visible from d_0, which is one hundred.

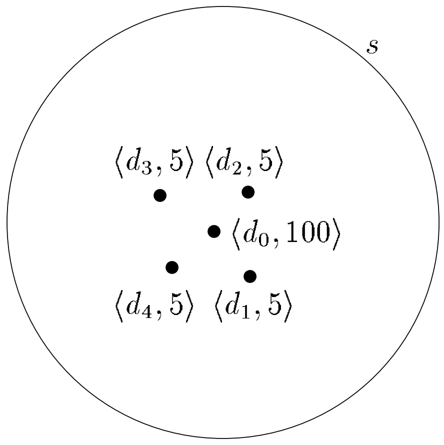

S
$\langle d_3, 5\rangle$ $\langle d_2, 5\rangle$
$\langle d_0, 100\rangle$
$\langle d_4, 5\rangle$ $\langle d_1, 5\rangle$

Figure 4.2: A d-i set for which a feasible polyhedron does not exist.

Similarly, for a particular d-i set S there may be feasible polyhedra only for some particular sizes. For example, a feasible polyhedron can never have size more than the sum of the integers in the d-i pairs of S assuming that the directions of S

are not within a half-space. Similarly, a feasible polyhedron can never have a size equal to an integer in S, simply because from a single direction not all faces can be visible.

Like in 2D, instead of S, the *proper* d-i set $\mathcal{S}$, which is derived from S, is used for all the results. The size N of a feasible polyhedron is the number its faces. Clearly, $N \geq 4$. The main problem is to create a feasible polyhedron of size N for $\mathcal{S}$.

4.2 Preliminaries

Throughout this chapter the surface of the sphere will be used quite often, so let us give some preliminaries on spherical geometry. Most of the contents of this section can be found in [46, 69, 78].

Let s be a sphere centered at the origin o. The *normal-point* of a face of a convex polyhedron P is the intersection points of the outward normal of f, translated to origin, with s. The (closed) hemisphere of s that is visible from a direction d is called the *visible hemisphere* of d. Observe that f is visible from a direction d if and only if the normal-point of f is strictly within the visible hemisphere of d. Also observe that P is bounded if and only if not all the normal-points of P intersect a single open hemisphere.

Two points p and q of s are called *antipodal* if they are the two opposite points of a diameter of s. Any region on s is considered as closed and connected. A point p a region r is said to be *strictly within* r if p is in the interior of r. A region r' is said to be *strictly within* r if every point of r' is strictly within r. A *spherical segment*, or simply a *segment*, between two points p and q in s is the shortest spherical arc of the great circle containing p and q. A region r of s is called *convex* if for any two non-antipodal points p and q of r the segment between p and q is also in r. Each of the four regions of s that are in between two planes passing through o is called a *spherical lune*, or simply a *lune*, with respect to the angle made by the two planes.

A *spherical polygon* with vertices $v_1, v_2 \ldots, v_n$ is a region within a single hemisphere and bounded by a closed path of ordered segments $\overline{v_1v_2}, \overline{v_1v_2} \ldots, \overline{v_nv_1}$ that does not intersect itself. The *opposite* of a spherical polygon x is the reflection of x through the origin.

4.3 Construction for directions in one plane

This section considers the case when all the directions lie in one plane. This case can be treated as a "warm up" for the more general cases. (An example of $\mathcal{S}$, where all directions are in one plane, and a corresponding feasible polyhedron was shown earlier in Figure 4.1). In this case a feasible polyhedron of a given size can be always constructed, if it exists. More interestingly, it is constructed from a feasible polygon. In fact, a stronger result can be achieved by showing that a feasible polygon can also be constructed from a feasible polyhedron. The construction algorithm takes $O(N \log N)$ and $O(N)$ time for the two cases, respectively, where N is the given size.

4.3.1 Basic definitions

Let $\mathcal{S} = \{\langle d_0, n_0\rangle, \langle d_1, n_1\rangle, \ldots, \langle d_{2K-1}, n_{2K-1}\rangle\}$ be the given proper d-i set. For any i, the visible hemisphere of d_i is represented by h_i. The region $\theta_i = h_{i+1} \backslash h_i$

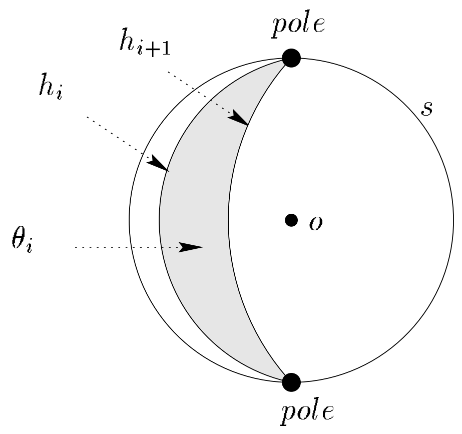


Figure 4.3: The i-th d-lune θ_i.

is called the *i-th d-lune* of $\mathcal{S}$. All d-lunes of $\mathcal{S}$ have two common antipodal points which are called *poles* of $\mathcal{S}$. See Figure 4.3. Observe that θ_i is non-empty and at most a hemisphere. Also observe that $h_i = \bigcup_{j=i-K}^{i-1} \theta_j$.

The d-lunes and visible hemispheres are 3D analogous of d-arcs and visible half-circles.

4.3.2 The construction

Let p be a feasible polygon of size N. The overall idea of creating a feasible polyhedron P from p is as follows. Consider the normal-points of the edges of p on s. Initially all of them are in the great circle c of s. Create P so that these normal-points are also its normal-points. But this current arrangement of normal-points will make P a cylinder with two ends unbounded. In order to make it bounded, slightly incline two of its faces in opposite directions by moving two normal-points towards the two poles of $\mathcal{S}$. The detail construction is the following theorem.

Theorem 4.1 *Given a proper d-i set $\mathcal{S}$ of size $2K$, where all the directions are in one plane and are ordered according to a circular-sequence of the directions, and given $N \geq 4$, then, a feasible polyhedron P for $\mathcal{S}$ of size N exists if and only if a feasible polygon p for $\mathcal{S}$ of size N exists. Moreover the time required to construct P from p is $O(N \log N)$ and p from P is $O(N)$.*

Proof. Constructing the polyhedron from the polygon.

Let $\mathcal{T}$ be the set of normal-points of p. They will finally be the normal-points for P. At this point all these normal-points intersect a single hemisphere bounded by c. Move two of these normal-points according to the following claim.

Claim 4.1 *It is possible to change the positions of two normal-points of $\mathcal{T}$ strictly within their respective d-lunes such that not all the normal-points of $\mathcal{T}$ intersect a single closed hemisphere any more. Moreover, the time required for such a movement is $O(N)$, where N is the size of $\mathcal{T}$.*

Proof. Choose four normal-points t_0, t_1, t_2 and t_3 from $\mathcal{T}$ so that their convex hull *strictly* includes the origin o. As the points of $\mathcal{T}$ do not intersect a single half-circle of c, four such points always exist. Let the four vectors from the origin to these four points be $\overrightarrow{t_0}$, $\overrightarrow{t_1}$, $\overrightarrow{t_2}$, and $\overrightarrow{t_3}$ respectively. Then, for some positive value of λ_0, λ_1, λ_2 and λ_3, the following equality holds,

$$\lambda_0 \overrightarrow{t_0} + \lambda_1 \overrightarrow{t_1} + \lambda_2 \overrightarrow{t_2} + \lambda_3 \overrightarrow{t_3} = 0. \qquad l1$$

Let in a circular sequence of the four normal-points be t_0, t_1, t_2, t_3. Let $\overrightarrow{n}$ be the normal vector of the plane of c. We will assign new position for t_0 and t_1. Let $\overrightarrow{t'_0} = \overrightarrow{t_0} + \lambda_1 \overrightarrow{n}$. Assign the new position of t_0 to t''_0, where the position of t''_0 is according to the vector $\overrightarrow{t''_0} = \overrightarrow{t'_0} \frac{\|t_0\|}{\|t'_0\|}$. (Here $\|x\|$ means the Euclidean length of

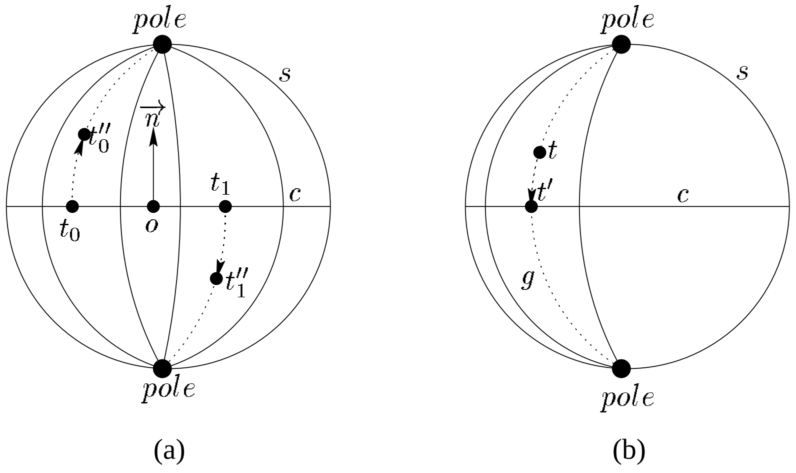


Figure 4.4: (a) Moving the normal-points t_0 and t_1 for creating the polyhedron from the polygon. (b) Move the normal-points t for creating the polygon from the polyhedron.

the vector $\overrightarrow{x}$). Similarly, let $\overrightarrow{t_1'} = \overrightarrow{t_1} - \lambda_0 \overrightarrow{n}$. Assign the new position of t_1 to t_1'', where $\overrightarrow{t_1''} = \overrightarrow{t_1'} \frac{||t_1||}{||t_1'||}$. This ends the movement.

Let us consider the justification now. The length of the new vectors $\overrightarrow{t_0''}$ and $\overrightarrow{t_1''}$ remain the same as that of $\overrightarrow{t_0}$ and $\overrightarrow{t_1}$, which is the radius of s. So t_0 and t_1 are on s. Moreover, t_0 and t_1 move to t_0'' and t_1'' due to the vector $\overrightarrow{n}$, which is perpendicular to both t_0 and t_1. This means the movement of t_0 and t_1 are along circular curves towards the poles. Therefore their movement are strictly within their respective d-lunes. See Figure 4.4(a).

We next prove that the four normal-points t_0'', t_1'', t_2, and t_3 no longer intersect a single hemisphere. For this, we first prove that t_0'', t_1'', t_2, and t_3 do not intersect a single great circle. t_2 and t_3 are not antipodal, because otherwise their line segment in the convex hull would intersect o, which would be a contradiction to the choice of t_0, t_1, t_2, and t_3. Since any two great circles intersect into antipodal points, any great circle passing through t_0'' and t_1'' cannot pass through t_2 and t_3. Therefore t_0'',

t''_1, t_2, and t_3 cannot be in a great circle.

With the above argument in hand, we now prove that t''_0, t''_1, t_2 and t_3 cannot intersect a single hemisphere. By way of contradiction assume that t''_0, t''_1, t_2, and t_3 intersect a single hemisphere h. Let $\overrightarrow{n_h}$ be the inner (i.e., inside h) normal vector of the plane of the great circle of h. Then for any positive value of λ''_0, λ''_1, λ''_2 and λ''_3, the following inequality of the dot product holds,

$$(\lambda''_0\overrightarrow{t''_0} + \lambda''_1\overrightarrow{t''_1} + \lambda''_2\overrightarrow{t_2} + \lambda''_3\overrightarrow{t_3}) \bullet \overrightarrow{n_h} \geq 0. \qquad l2$$

Disprove ($l2$) will eiyld the proof of the claim. Choose $\lambda''_0 = \lambda_0\frac{||t'_0||}{||t_0||}$, $\lambda''_1 = \lambda_1\frac{||t'_1||}{||t_1||}$, $\lambda''_2 = \lambda_2$, and $\lambda''_3 = \lambda_3$, which are all positive. By (l_1),

$$\lambda''_0\overrightarrow{t''_0} + \lambda''_1\overrightarrow{t''_1} + \lambda''_2\overrightarrow{t_2} + \lambda''_3\overrightarrow{t_3} = \lambda_0(\overrightarrow{t_0} + \lambda_1\overrightarrow{n}) + \lambda_1(\overrightarrow{t_1} - \lambda_0\overrightarrow{n}) + \lambda_2\overrightarrow{t_2} + \lambda_3\overrightarrow{t_3} = 0.$$

This means that if ($l2$) holds, then it holds only for the equality. But since t''_0, t''_1, t_2, and t_3 intersect a single hemisphere, the equality in ($l2$) holds only when the vectors of t''_0, t''_1, t_2, and t_3 are coplanar, i.e., t''_0, t''_1, t_2, and t_3 are in a great circle, which was already proved to be impossible.

For time complexity, the only place that may require more than constant time is to select the four points from $\mathcal{T}$. For this simply take any three consecutive points and select the fourth one by going through the remaining points of $\mathcal{T}$ in $O(N)$ time. □

With the above claim, not all N normal-points that do not intersect a single hemisphere of s. Create P by taking the intersection of the half-spaces defined by the tangent planes through these normal-points and containing s. This ends the construction of the polyhedron. See Figure 4.5 for a possible example of P and p corresponding to each other, where the arrows show the ways through which the normal-points of two edges of p have been lifted for the construction of P.

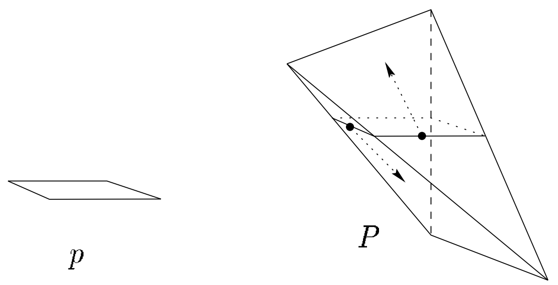


Figure 4.5: A possible example of P and p of size four that correspond to each other.

Since the points of $\mathcal{T}$ remain in their respective d-lunes, we continue to see n_i normal-points from d_i, hence the construction is correct. Claim 4.1 takes $O(N)$ time. Finally, creating P by taking the intersection of the half-spaces takes $O(N \log N)$ time [68]. Therefore the total time is $O(N \log N)$.

Constructing the polygon from the polyhedron.

First find the normal-points of the faces of P. Let their set be $\mathcal{T}$. For each point t in $\mathcal{T}$, then do the following. If t is in c, then leave it as it is. Otherwise let g be the great circle passing through t and the poles. Let t be strictly within the d-lune θ_i. Let t' be the intersection of g and c. Move t to t'. See Figure 4.4(b). Finally, construct p from these normal-points in $O(N)$ time, as mentioned in Section 3.3.4. This ends the construction of p.

Since P is feasible, the points of $\mathcal{T}$ are not in the boundary of the d-lunes, and h_i contains n_i points strictly within it. By the above movement and since $H_i = h_i \cap c$, H_i has those n_i points strictly within it too, and so d_i sees n_i edges of p. $\square$

After the above theorem the necessary and sufficient conditions for the existence of a feasible polyhedron of size N for $\mathcal{S}$ is the same as it is for a feasible polygon of size N for $\mathcal{S}$. Moreover, the maximum and minimum size of feasible polyhedra are the same and can be computed similarly as for feasible polygons.

Corollary 4.1 *Given a proper d-i set $\mathcal{S}$ of size $2K$, where all directions are co-planar and the d-i pairs are ordered according to a circular sequence of the directions, and given $N \geq 4$. Then,*

- *The necessary and sufficient conditions for the existence of a feasible polygon P of size N is $D \leq N$.*
- *The maximum and minimum size of P can be found in $O(K + v \log v)$ time, where v is the number original d-i pairs of $\mathcal{S}$ whose corresponding next d-i pairs are derived.*

Proof. Follows when Theorem 4.1 is combined with Theorem 3.1 and Theorem 3.3. □

4.4 Construction for directions in two planes

This section considers the case when the directions of $\mathcal{S}$ are covered by two planes through the origin. Note that the planes are not unique and are given. The main result in this section are the followings. When all directions see at least four faces, we can construct a feasible polyhedron, when it exists. When the directions see arbitrary number of faces, we can construct a feasible polyhedron, if it exists, except for one sub-case when the maximum among the two sums of the view differences for the directions in two planes is N.

The construction processes are much more complicated than earlier and for better understanding, a road map is presented before we start.

Road map

We will start with some basic definitions in Section 4.4.1. It is obvious that the existence of a feasible polyhedron for each plane of directions is necesary and one can check the existence for indivual planes as described in the previous section. But we will see in Section 4.4.2 that this is not sufficient. The construction starts with an outline in Section 4.4.3. The whole construction process is a sequence of two phases: finding a "valid assignment" and then finding a "valid selection" of the normal-points (these terms are defined later). A valid assignment will ensures that each direction see required number of faces and a valid selection will ensure that the polyhedron is bounded. Finding a valid assignment is divided into two cases based on whether two planes have common directions or not and they are in Sections 4.4.4 and 4.4.5, respectively. Finding a valid selection is also divided into two cases based on whether all directions see at least four faces or not. The case when the directions see arbitrary number of faces is handled in Section 4.4.6. When all directions see at least four faces, the construction is in Section 4.4.7. (Construction is very lengthy here). Finally, Section 4.4.8 shows by an example why the technique of finding a valid assignment and then a valid selection does not always work when the directions see arbitrary number of faces.

4.4.1 Basic definitions

When the directions of $\mathcal{S}$ are in two planes (or in general, more than one plane), the arrangement of the d-lunes becomes more complicated, There are two distinct cases: (1) two planes have a pair of opposite directions in common (note that two planes can have exactly one pair of opposite direction in common, and (2) no common directions among the planes. For example, see Figure 4.6(a), where the directions drawn bold and very small are common to both of them. In the case of a common pair of directions, the arrangement of the d-lunes is relatively simple. The poles are on the bounding great circle of the two visible hemispheres of the two common directions. The d-lunes and their arrangement are partitioned into two by these two hemispheres. See Figure 4.6(b) and (c) for the two cases respectively.

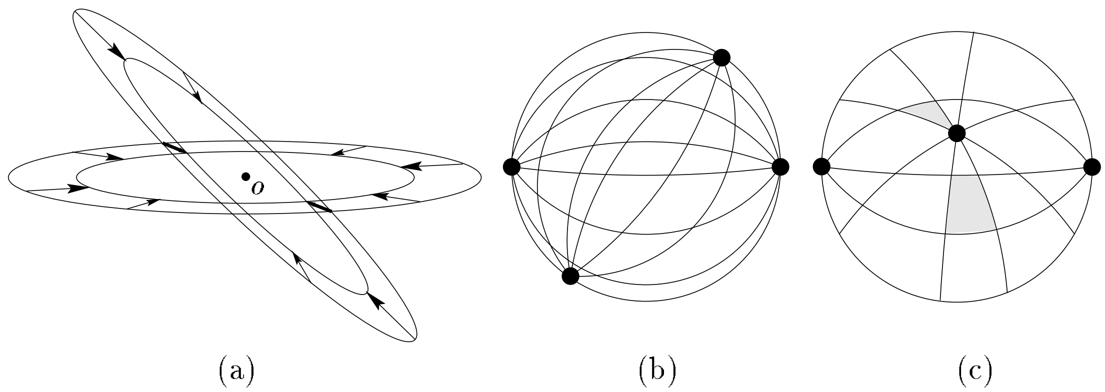


Figure 4.6: Arrangement of d-lunes when directions in two planes. (a) Example of $\mathcal{S}$ with two planes having a common d-i pair of opposite directions. (b) Arrangement of the d-lunes when such a common d-i pair of opposite directions exists. (c) Arrangement of the d-lunes when no such common d-i pair exists; also in this figure the shaded d-polygons have size three and four respectively.

The arrangement of the d-lunes divides s into spherical polygons called *d-polygons*. The sizes of these spherical polygons depend upon the number of planes. When the number of planes is two, then the d-lunes from two planes create s-polygons of sizes three and four. For example see Figure 4.6(b), where $\mathcal{S}$ has two planes of directions and two d-polygons of size three and four are shown shaded.

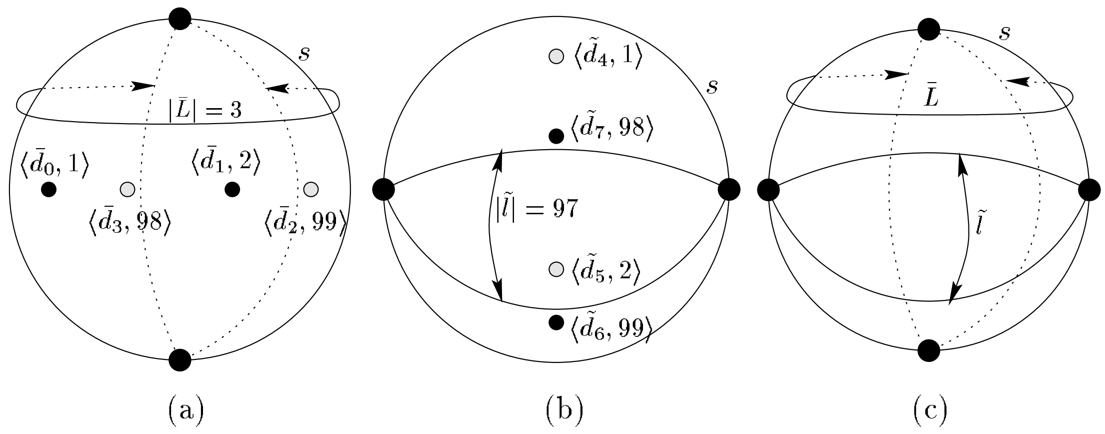
Since the d-lunes are non-empty and at most a hemisphere and since a d-polygon is the intersection of more than one d-lune from different planes, a d-polygon is non-enpty and strictly less than hemisphere. For each d-polygon θ there is an opposite d-polygon θ' which is the intersection of the d-lunes opposite to the d-lunes intersecting to θ. A d-polygon resulting from the intersection of i-th and j-th d-lunes from the two planes of directions, respectively, is denotd by $\theta_{i,j}$.

4.4.2 Insufficiency of individual feasibility

Let $\bar{\mathcal{S}}$ and $\tilde{\mathcal{S}}$ be the two proper d-i sets corresponding to two planes of directions ($\bar{}$ and $\tilde{}$ are used over the notations that are related to $\bar{\mathcal{S}}$ and $\tilde{\mathcal{S}}$, respectively). So $\mathcal{S} = \bar{\mathcal{S}} \cup \tilde{\mathcal{S}}$. To construct a polyhedron for $\mathcal{S}$ it is necessary that each of $\bar{\mathcal{S}}$ and $\bar{\mathcal{S}}$ has a feasible polyhedron separately. But it is not sufficient, because there may be two separate feasible polyhedra for $\bar{\mathcal{S}}$ and $\tilde{\mathcal{S}}$ but no single feasible polyhedron for $\mathcal{S}$. This is shown in the following example.

Let $\bar{\mathcal{S}} = \{\langle \bar{d}_0, 1\rangle, \langle \bar{d}_1, 2\rangle, \langle \bar{d}_2, 99\rangle, \langle \bar{d}_3, 98\rangle\}$ and $\tilde{\mathcal{S}} = \{\langle \tilde{d}_4, 1\rangle, \langle \tilde{d}_5, 2\rangle, \langle \tilde{d}_6, 99\rangle, \langle \tilde{d}_7, 98\rangle\}$ in Figure 4.7(a) and (b) respectively. Here N is one hundred. $\bar{\mathcal{S}}$ and $\tilde{\mathcal{S}}$ are the same except that the two planes are perpendicular. For both sets the view differences are 1, 97, 0 and 0, and so their sum is less than N. From Corollary 4.1 a feasible polyhedron exists for both of them.

Now, for the two d-i pairs $\langle \bar{d}_0, 1\rangle$ and $\langle \bar{d}_1, 2\rangle$ of $\bar{\mathcal{S}}$ the total number of visible faces is three. The union $\bar{L}$ of the visible hemispheres $\bar{h}_0$ and $\bar{h}_1$ contains exactly three normal-points (of these three faces) strictly within it. The remaining lune $\bar{l}$ contains exactly ninety-seven normal-points strictly within it. See Figure 4.7(a). Similarly, the two lunes $\tilde{L}$ and $\tilde{l}$ for the second set contain three and ninety-seven normal-points strictly within it. See Figure 4.7(b). But when $\bar{\mathcal{S}}$ and $\tilde{\mathcal{S}}$ are combined together, because of the combination of the two sets of directions, $\tilde{l}$ is a subset of $\bar{L}$. See Figure 4.7(c). So it is not possible to select normal-points so that $\tilde{l}$ gets ninety-seven of them and $\bar{L}$ gets three of them at the same time.



s
$|\bar{L}| = 3$
$\langle \bar{d}_0, 1\rangle$
$\langle \bar{d}_1, 2\rangle$
$\langle \bar{d}_3, 98\rangle$
$\langle \bar{d}_2, 99\rangle$
(a)
$\langle \tilde{d}_4, 1\rangle$
s
$\langle \tilde{d}_7, 98\rangle$
$|\tilde{l}| = 97$
$\langle \tilde{d}_5, 2\rangle$
$\langle \tilde{d}_6, 99\rangle$
(b)
s
$\bar{L}$
$\tilde{l}$
(c)

Figure 4.7: (a) $\bar{\mathcal{S}}$, $\bar{l}$ and $\bar{L}$; $|\bar{l}| = 97$ and $|\bar{L}| = 3$. (b) $\tilde{\mathcal{S}}$, $\tilde{l}$ and $\tilde{L}$; $|\tilde{l}| = 97$ and $|\tilde{L}| = 3$. (c) When $\bar{\mathcal{S}}$ and $\tilde{\mathcal{S}}$ are combined together, $\tilde{l} \subset \bar{L}$.

Observation 4.1 *The existence of a feasible polyhedron for each plane of directions of $\mathcal{S}$ is necessary but not sufficient for the existence of a feasible polyhedron for $\mathcal{S}$.*

Since the existence of a feasible polyhedron for individual planes is necessary, for the rest of this section we will assume that for each of $\bar{\mathcal{S}}$ and $\tilde{\mathcal{S}}$ $\bar{D} \leq N$ and $\tilde{D} \leq N$.

4.4.3 Construction outline

Let us go back to the example of the previous section. The reason that a feasible polyhedron does not exist in that example is that one cannot simultaneously assign normal-points to $\bar{L}$ and $\tilde{l}$ according to their required number, which in turn cannot simultaneously satisfy the required number of normal-points on the visible hemisphere of all directions. So we need to find a "distribution" of the normal-points among the d-polygons so that all d-lunes as well as all hemispheres get their required values. More formally, we need to assign N normal-points to the d-polygons such that for each d-i pair $\langle d, n\rangle$ in $\bar{\mathcal{S}}$ or $\tilde{\mathcal{S}}$ the number of points in the visible hemisphere of d is n. Such an assignment to the d-polygons is called *a valid assignment*, and the specific number $\Delta_{i,j}$ assigned to d-polygon $\theta_{i,j}$ the *final view difference* of $\theta_{i,j}$. We will see that it is always possible to find a valid assignment when it exists. Note that there may be more than one valid assignment.

As already indicated earlier, a valid assignment only ensures that each direction will see corresponding number of normal-points despite how we choose the actual positions of the normal-points. So if the feasible polyhedron is allowed to be unbounded, then the existence of a valid assignment is necessary and sufficient for the existence of a feasible polyhedron.

For a bounded polyhedron, the actual positions of the normal-points are important— the feasible polyhedron is bounded if and only if not all normal-points intersect a single closed hemisphere. So on top of a valid assignment, we need a *valid selection*. Formally, a valid selection is the process of selecting the positions of normal-points on s so that the number of normal-points that each d-polygon gets is equal to its final view difference (as assigned by the valid assignment) and that not all normal-points intersect a single hemisphere.

The following observation formally states that the existence of a valid assignment and a valid selection is necessary and also sufficient for the existence of a feasible polyhedron. The implication of this observation in 2D is that a valid assignment always implies a valid selection, and that is why it was possible to create a feasible polygon as long as $D \leq N$ (see Chapter 3).

Observation 4.2 *Given $\mathcal{S}$, the existence of a valid assignment of a total value of N among the d-polygons of $\mathcal{S}$ and a valid selection of N normal-points on s is necessary and sufficient to create, when possible, a feasible polyhedron P of size N for $\mathcal{S}$.*

We will quite frequently use the term "the sum of the final view differences of the d-polygons" of a particular d-lune or visible hemisphere, and they are called *poly-sum* for short.

A helper lemma

Earlier in Section 4.4.1, we saw that the arrangement of the d-lunes of $\bar{\mathcal{S}}$ and $\tilde{\mathcal{S}}$ is similar to a collection of two dimensional matrices where each spherical polygon corresponds to a cell of a matrix. For finding a valid assignment we have to assign the final view differences of spherical polygons, and we will do that through assigning numbers to the corresponding matrix cells. In the following we present an auxiliary lemma which in general assign numbers in a matrix.

The general version of this lemma, where assigning a value to a particular location has some cost function and it is required to minimize the total cost, is known as the *transportation problem* [8, 21, 42, 47, 64]. A similar problem, called *latin squares*, is to assign a $n \times n$ matrix where each row and column is a permutation of the integers from 1 to n [7, 85].

Lemma 4.1 *Let R and C be two arrays of non-negative integers of size m and n, respectively, such that $\sum_{i=1}^{m} R_i \leq \sum_{j=1}^{n} C_j$. Then, it is always possible to assign non-negative integers $M_{i,j}$ with $1 \leq i \leq m$ and $1 \leq j \leq n$ such that $\sum_{j=1}^{n} M_{i,j} = R_i$ and $\sum_{i=1}^{m} M_{i,j} \leq C_j$. Moreover, the time required is $O(m+n)$ if the outputs are only the elements that have been assigned and all other elements are zero.*

Proof.

The following pseudocode will assign the elements of M.

```
for $i = 1 \ldots m$, let $t_i = R_i$
for $j = 1 \ldots n$, let $u_j = C_j$          // $t_i$ and $u_j$ are temporary variables
let $j = 1$
for $i = 1 \ldots m$
    while $t_i > 0$
        $M_{i,j} = \min\{t_i, u_j\}$
        $t_i = t_i - M_{i,j}$
        $u_j = u_j - M_{i,j}$
       if $u_j = 0$, then $j = j + 1$
```

Let us now consider the justification. $\sum_{i=1}^{m} R_i \leq \sum_{j=1}^{n} C_j$ implies that for any i, $R_i \leq \sum_{j=1}^{n} C_j$. Since R_i intersects all columns, its elements can be assigned a maximum value of $\sum_{j=1}^{n} C_j$ in total. So from the algorithm, the elements of the first row of M are assigned a total value of R_1. After assigning the first row the elements of second row can be assigned a total of $\sum_{j=1}^{n} C_j - R_1$, which is at least R_2. So the elements of the second row are assigned a total value of R_2. In this way for all i, the elements of i-th row get a total value of R_i. Therefore the sum of all elements of M is $\sum_{i=1}^{m} R_i$.

For the columns of M, for all j we keep track of the total value assigned to the j-th column by checking u_j to be zero. Therefore the j-th column is assigned a total value of no more than the initial value of u_j, which is C_j.

Finally, we increment j only if u_j becomes zero. Since $\sum_{i=1}^{m} R_i \leq \sum_{j=1}^{n} C_j$, u_j becomes zero at most n times before the assignment is complete, so during the assignment the value of j does not exceed n.

If the outputs are only the elements that have been assigned and all other elements are zero, then the time complexity is clearly $O(m+n)$. $\square$

4.4.4 Valid assignment when planes have common directions

We will now show that it is always possible to find a valid assignment of the final view differences of the d-polygons when $\bar{\mathcal{S}}$ and $\tilde{\mathcal{S}}$ have common d-i pairs. In addition to that, at the same time it is also possible to find two opposite positive d-polygons. This latter work is part of the next step of valid selection but is being done here in advance as a relevant work.

Lemma 4.2 *Given $\bar{\mathcal{S}}$ and $\tilde{\mathcal{S}}$, where $\bar{\mathcal{S}}$ and $\tilde{\mathcal{S}}$ have a common d-i pair of opposite d-i pairs and $\bar{D}, \tilde{D} \leq N$, then, it is always possible in $O(\bar{K}+\tilde{K})$ time to assign the final view differences of the d-polygons such that for any d-i pair $\langle d, n\rangle$ of $\bar{\mathcal{S}}$ or $\tilde{\mathcal{S}}$, the poly-sum of the visible hemisphere of d is n. Moreover, if* $\mathbf{max}\{\bar{D}, \tilde{D}\} < N$*, then it is possible within the same time to find two opposite positive d-polygons.*

Proof. The overall idea of the proof is as follows. We will first see how the d-lunes of $\bar{\mathcal{S}}$ and $\tilde{\mathcal{S}}$ are arranged in this case. Then the final view differences of the d-polygons are assigned through assigning the final view differences of the d-lunes. Initially a total value of $\mathbf{max}\{\bar{D}, \tilde{D}\}$ will be assignmed, then if $\mathbf{max}\{\bar{D}, \tilde{D}\} < N$, the remaining value of $N - \mathbf{max}\{\bar{D}, \tilde{D}\}$ are assigned to two opposite d-polygons.

We now describe the process in detail. Without loss of generality let the d-i pair of opposite d-i pairs of $\bar{\mathcal{S}}$ that are common to $\tilde{\mathcal{S}}$ be $\langle \bar{d}_0, \bar{n}_0\rangle$ and $\langle \bar{d}_{\bar{K}}, \bar{n}_{\bar{K}}\rangle$. So the great circle of the visible hemispheres $\bar{h}_0$ and $\bar{h}_{\bar{K}}$ passes through the poles of $\bar{\mathcal{S}}$

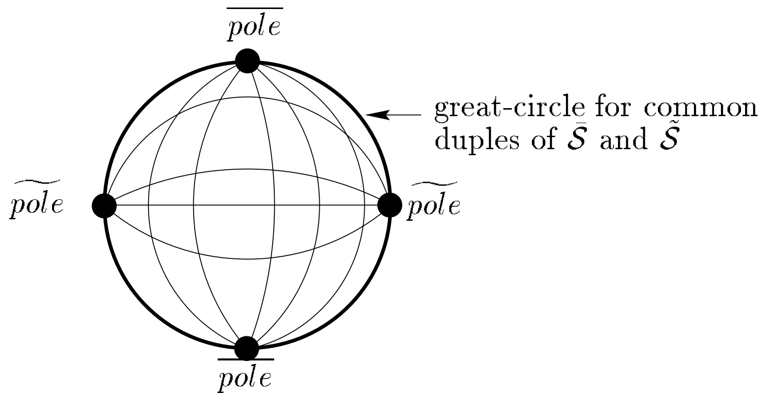

$\overline{pole}$
great-circle for common duples of $\bar{\mathcal{S}}$ and $\tilde{\mathcal{S}}$
$\widetilde{pole}$
$\widetilde{pole}$
$\overline{pole}$

Figure 4.8: The d-i pair $\langle \bar{d}_0, \bar{h}_0 \rangle$ of $\bar{\mathcal{S}}$ is common to $\tilde{\mathcal{S}}$ and for that the great circle corresponding to $\bar{h}_0$ passes through all four poles.

and $\tilde{\mathcal{S}}$. This great circle also divides the d-polygons into two similar sets — one in $\bar{h}_0$ and the other one in $\bar{h}_{\bar{K}}$. (See Figure 4.8).

We start by assigning a total value of $\mathbf{max}\{\bar{D}, \tilde{D}\}$ to the final view differences of the d-lunes. First consider the d-lunes of $\bar{\mathcal{S}}$. For all i, set $\bar{\Delta}_i$ as $\bar{\delta}_i$. If $\bar{D} \neq \tilde{D}$, without loss of generality assume that $\bar{D} < \tilde{D}$. Then choose an arbitrary d-i pair of opposite d-lunes and increase both of their corresponding final view differences by $\frac{1}{2}(\tilde{D} - \bar{D})$. (By Lemma 3.2 in Chapter 3, both $\frac{1}{2}(N - \bar{D})$ and $\frac{1}{2}(N - \tilde{D})$ are even, so $\frac{1}{2}(\tilde{D} - \bar{D})$ is also even). For the d-lunes of $\tilde{\mathcal{S}}$, for all j, set $\tilde{\Delta}_j$ to $\tilde{\delta}_j$.

We now assign the final view differences of the d-polygons. We will describe how to assign the d-polygons of $\bar{h}_0$ only. Assigning the d-polygons of $\bar{h}_{\bar{K}}$ is similar.

Without loss of generality assume that $\bar{h}_0$ contains the d-lunes from $\tilde{\theta}_0$ from $\tilde{\theta}_{\tilde{K}}$ of $\tilde{\mathcal{S}}$. In $\bar{h}_0$ all the d-lunes of $\bar{\mathcal{S}}$ intersect all d-lunes of $\tilde{\mathcal{S}}$. This intersection forms a $\bar{K} \times \tilde{K}$ matrix of d-polygons. See Figure 4.8. Apply Lemma 4.1 to get the values of the final view differences of these d-polygons as follows. The d-lunes from $\bar{\mathcal{S}}$ and $\tilde{\mathcal{S}}$ correspond to the rows and columns of M respectively. For $0 \leq i \leq \bar{K} - 1$, set $R_i = \bar{\Delta}_i$, and for $0 \leq j \leq \tilde{K} - 1$, set $C_j = \tilde{\Delta}_j$. In return from this lemma, the value of each cell $M_{i,j}$ gives the value of $\Delta_{i,j}$. Similarly assign the d-polygons of $h_{\bar{K}}$.

Finally, if $\tilde{D} < N$, then choose the two arbitrary opposite d-polygons and increase both of them by $\frac{1}{2}(N - \tilde{D})$. At the same time, also adjust the final view difference of their corresponding d-lunes by increasing each of them by the same amount $\frac{1}{2}(N - \tilde{D})$. This ends the assignment.

Justification. Let us now consider the justification. We first consider that the assignment is a valid one. Lemma 4.1, after being applied, confirms that for any d-lune, the poly-sum is same as its final view difference. Moreover, this equality holds for the case when $\tilde{D} < N$, since we increased the value of two opposite d-polygons and adjusted the value of all corresponding d-lunes.

Now consider an arbitrary d-i pair $\langle \bar{d}_i, \bar{n}_i \rangle$ from $\bar{\mathcal{S}}$. Since $\bar{h}_i = \bigcup_{l=i+\bar{K}}^{i-1} \bar{\theta}_l$, according to the algorithm the poly-sum of $\bar{h}_i$ is $\sum_{l=i+\bar{K}}^{i-1} \bar{\Delta}_l$. We need to prove that this value is $\bar{n}_i$. Observe that among any two opposite d-lunes exactly one is in $\bar{h}_i$. So $\sum_{l=i+\bar{K}}^{i-1} \bar{\Delta}_l = \sum_{l=i+\bar{K}}^{i-1} \bar{\delta}_l + \frac{1}{2}(\tilde{D} - \bar{D}) + \frac{1}{2}(N - \tilde{D})$, where the last two terms come from the increased value of d-lunes in the two cases when $\bar{D} < \tilde{D}$ and $\tilde{D} < N$ respectively. After simplification, this value becomes $\sum_{l=i+\bar{K}}^{i-1} \bar{\delta}_l + \frac{1}{2}(N - \bar{D})$, which according to Lemma 3.2 is $\bar{n}_i$.

Similarly, consider an arbitrary d-i pair $\langle \tilde{d}_j, \tilde{n}_j \rangle$ from $\tilde{\mathcal{S}}$. Since $\tilde{h}_j = \bigcup_{l=j+\tilde{K}}^{j-1} \tilde{\theta}_l$ and thus the poly-sum of $\tilde{h}_j$ is $\sum_{l=j+\tilde{K}}^{j-1} \tilde{\Delta}_l$. We need to prove that this value is $\tilde{n}_j$. By an argument similar to that in the previous paragraph, $\sum_{l=j+\tilde{K}}^{j-1} \tilde{\Delta}_l = \sum_{l=j+\tilde{K}}^{j-1} \tilde{\delta}_l + \frac{1}{2}(N - \tilde{D})$, where the last term comes from the increased value of the d-lunes in the case when $\tilde{D} < N$. But from Lemma 3.2, this value is $\tilde{n}_j$. So the assignment is valid.

Finally, for the last part of the lemma observe that the two opposite d-polygons $\theta_{c,d}$ and $\theta_{c+\bar{K},d+\bar{K}}$ have (possibly the same) positive view differences.

For the time complexity. the initial assignment of the final view differences of the d-lunes of $\bar{S}$ and $\tilde{S}$ takes $O(\bar{K}+\tilde{K})$ time. While applying Lemma 4.1 for the d-polygons of $\bar{h}_0$ and $\bar{h}_{\bar{K}}$, the size of the matrix is $\bar{K}\times\tilde{K}$ and thus it takes $O(\bar{K}+\tilde{K})$ time for both cases. Finally, when $\tilde{D}<N$, all the extra work takes constant time. So the total time is $O(\bar{K}+\tilde{K})$. $\square$

4.4.5 Valid assignment when no common direction exists

The following lemma will find a valid assignment, when it exists, for the case when $\bar{S}$ and $\tilde{S}$ have no common d-i pairs. Similarly to the previous case, whenever a valid assignment exists we will also find two opposite positive d-polygons in the same lemma.

Lemma 4.3 *Given $\bar{S}$ and $\tilde{S}$, where $\bar{S}$ and $\tilde{S}$ have no common d-i pair of opposite d-i pairs and $\bar{D},\tilde{D}\leq N$, then, it is possible to assign final view differences of the d-polygons, if it exists, in $O(\bar{K}+\tilde{K})$ time such that for any d-i pair $\langle d,n\rangle$ of $\bar{S}$ or $\tilde{S}$, the poly-sum of the visible hemisphere of d is n. Moreover, if $\mathbf{max}\{\bar{D},\tilde{D}\}<N$, then it is possible within the same time to find two positive opposite d-polygons.*

Proof. The overall idea of the proof is as follows. We will first see how the d-lunes of $\bar{S}$ and $\tilde{S}$ are arranged in this case. Then we will give necessary and sufficient conditions for the existence of a valid assignment. The proof for this condition is constructive and will give an algorithm for the valid assignment. It will also ensure that there are two opposite d-polygons when $\mathbf{max}\{\bar{D},\tilde{D}\}<N$.

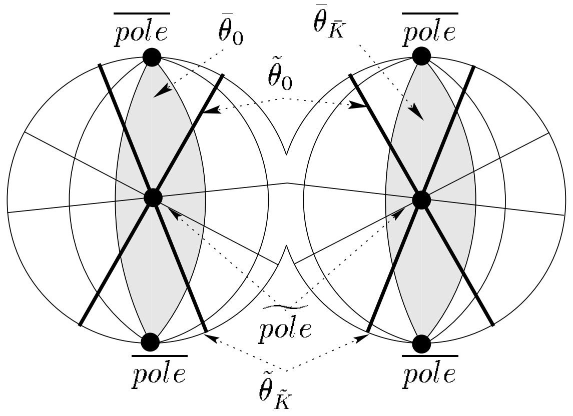


Figure 4.9: The two opposite views of the arrangement of the d-lunes when $\bar{\mathcal{S}}$ and $\tilde{\mathcal{S}}$ do not have any common d-i pair. $\bar{\theta}_0$ and $\theta_{\bar{K}}$ (the lightly shaded d-lunes) contain two poles of $\tilde{\mathcal{S}}$ and $\tilde{\theta}_0$ and $\theta_{\tilde{K}}$ (shown with their boundary in bold) contain the two poles of $\bar{\mathcal{S}}$.

Let us now go through the details. As $\bar{\mathcal{S}}$ and $\tilde{\mathcal{S}}$ do not have any common d-i pair, the two poles of $\bar{\mathcal{S}}$ (similarly $\tilde{\mathcal{S}}$) are strictly within two opposite d-lunes of $\tilde{\mathcal{S}}$ (similarly $\bar{\mathcal{S}}$). Without loss of generality let these two d-lunes of $\bar{\mathcal{S}}$ be $\bar{\theta}_0$ and $\bar{\theta}_{\bar{K}}$ and two d-lunes of $\tilde{\mathcal{S}}$ be $\tilde{\theta}_0$ and $\tilde{\theta}_{\tilde{K}}$. Observe that both $\bar{\theta}_0$ and $\bar{\theta}_{\bar{K}}$ intersect all d-lunes of $\tilde{\mathcal{S}}$ and similarly both $\tilde{\theta}_0$ and $\tilde{\theta}_{\tilde{K}}$ intersect all d-lunes of $\bar{\mathcal{S}}$. (See Figure 4.9).

Let $\bar{W}_1$ and $\bar{W}_2$ be the two sets of d-lunes that are in between $\bar{\theta}_0$ and $\bar{\theta}_{\bar{K}}$ and let $\bar{X}_1$ and $\bar{X}_2$ be the sum of the view differences of the d-lunes of $\bar{W}_1$ and $\bar{W}_2$ respectively. More formally,

$$\bar{W}_1 = \{\bar{\theta}_1, \bar{\theta}_2, \ldots, \bar{\theta}_{\bar{K}-1}\}, \bar{W}_2 = \{\bar{\theta}_{\bar{K}+1}, \bar{\theta}_{\bar{K}+2}, \ldots, \bar{\theta}_{2\bar{K}-1}\},$$

$$\bar{X}_1 = \sum_{i=1}^{\bar{K}-1} \bar{\delta}_i, \text{ and } \bar{X}_2 = \sum_{i=\bar{K}+1}^{2\bar{K}-1} \bar{\delta}_i.$$

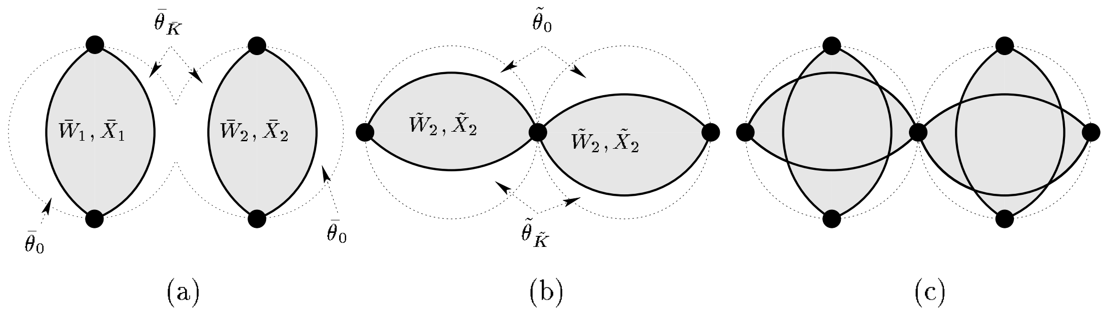


Figure 4.10: (a) $\bar W_1$, $\bar X_1$, $\bar W_2$ and $\bar X_2$ by two opposite views of s. (b) $\tilde W_1$, $\tilde X_1$, $\tilde W_2$ and $\tilde X_2$ by two opposite views of s for each. (c) The two opposite views of s when $\bar W_1$, $\bar W_2$, $\tilde W_1$ and $\tilde W_2$ are combined together.

For example, for $\bar{\mathcal{S}} = \{\langle \bar d_0, 25\rangle, \langle \bar d_0, 50\rangle, \langle \bar d_0, 85\rangle, \langle \bar d_0, 75\rangle, \langle \bar d_0, 50\rangle, \langle \bar d_0, 15\rangle\}$, $\bar W_1 = \{\bar\theta_1\}$, $\bar W_2 = \{\bar\theta_4\}$, $\bar X_1 = 60$, and $\bar X_2 = 10$. Similarly, let

$$\tilde W_1 = \{\tilde\theta_1, \tilde\theta_2, \ldots, \tilde\theta_{\tilde K - 1}\}, \tilde W_2 = \{\tilde\theta_{\tilde K+1}, \tilde\theta_{\tilde K+2}, \ldots, \tilde\theta_{2\tilde K-1}\},$$

$$\tilde X_1 = \sum_{j=1}^{\tilde K-1} \tilde\delta_j, \text{ and } \tilde X_2 = \sum_{j=\tilde K+1}^{2\tilde K-1} \tilde\delta_j.$$

Without loss of generality assume that all d-lunes of $\bar W_1$ (similarly $\bar W_2$) intersect all d-lunes of $\tilde W_1$ (respectively $\tilde W_2$). (See Figure 4.10).

Claim 4.2 *There exists a valid assignment for the final view differences of the d-polygons if and only if the following two conditions are satisfied.*

$$\bar X_1 - \mathbf{min}\{\bar X_1, \tilde X_1\} + \bar X_2 - \mathbf{min}\{\bar X_2, \tilde X_2\} \le \tilde\delta_0 + \tilde\delta_{\tilde K} + (N - \tilde D) \qquad (c1)$$

$$\tilde X_1 - \mathbf{min}\{\bar X_1, \tilde X_1\} + \tilde X_2 - \mathbf{min}\{\bar X_2, \tilde X_2\} \le \bar\delta_0 + \bar\delta_{\bar K} + (N - \bar D) \qquad (c2)$$

Proof. ($\Longrightarrow$) There are four combinations for the left hand side of $(c1)$, and for each of them we will show that $(c1)$ holds.

Case (i): $\bar X_1 \le \tilde X_1$ and $\bar X_2 \le \tilde X_2$. In this case the left hand side of $(c1)$ is 0. Since $\tilde\delta_0$, $\tilde\delta_{\tilde K}$, and $(N - \tilde D)$ are non-negative, $(c1)$ holds.

Case (ii): $\bar{X}_1 > \tilde{X}_1$ and $\bar{X}_2 \leq \tilde{X}_2$. In this case the left hand side of ($c1$) is $\bar{X}_1 - \tilde{X}_1$. For any valid assignment, the number of normal-points in $\bar{W}_1$, which is at least $\bar{X}_1$, is less than the number of normal-points in $\tilde{W}_1 \cup \tilde{\theta}_0 \cup \tilde{\theta}_{\tilde{K}}$. But the number of normal-points in $\tilde{W}_2 \cup \tilde{\theta}_0$ is $\tilde{n}_0$, which is, by Lemma 3.2, $\tilde{X}_1 + \tilde{\delta}_0 + \frac{1}{2}(N - \tilde{D})$. Therefore, $\bar{X}_1 \leq \tilde{X}_1 + \tilde{\delta}_0 + \frac{1}{2}(N - \tilde{D})$, so ($c1$) holds.

Case (iii): $\bar{X}_1 \leq \tilde{X}_1$ and $\bar{X}_2 > \tilde{X}_2$. The proof for this case is analogous to that for Case (ii) above.

Case (iv): $\bar{X}_1 > \tilde{X}_1$ and $\bar{X}_2 > \tilde{X}_2$. Here the left hand side of ($c1$) is $(\bar{X}_1 - \tilde{X}_1) + (\bar{X}_2 - \tilde{X}_2)$. In Case (ii) and Case (iii) we have seen that for any valid assignment, $\bar{X}_1 \leq \tilde{X}_1 + \tilde{\delta}_0 + \frac{1}{2}(N - \tilde{D})$, and $\bar{X}_2 \leq \tilde{X}_2 + \tilde{\delta}_{\tilde{K}} + \frac{1}{2}(N - \tilde{D})$. Therefore, $(\bar{X}_1 - \tilde{X}_1) + (\bar{X}_2 - \tilde{X}_2) \leq \tilde{\delta}_0 + \tilde{\delta}_{\tilde{K}} + (N - \tilde{D})$.

($\Longleftarrow$) The proof is constructive here— when the conditions are satisfied, a valid assignment can be found as required by the lemma. First assign the final view differences of the d-lunes. Without loss of generality assume that, when $\bar{D} \neq \tilde{D}$, $\bar{D} < \tilde{D}$. For all i, set $\bar{\Delta}_i = \bar{\delta}_i$. If $\bar{D} < N$, then increase both $\bar{\Delta}_0$ and $\bar{\Delta}_{\bar{K}}$ by $\frac{1}{2}(N - \bar{D})$. Similarly, for all j, set $\tilde{\Delta}_j$ to $\tilde{\delta}_j$. If $\tilde{D} < N$, then increase both $\bar{\Delta}_0$ and $\bar{\Delta}_{\tilde{K}}$ by $\frac{1}{2}(N - \tilde{D})$.

Next assign the final view differences of the d-polygons. This assignment is divided into three steps. In Step 1 assign the d-polygons that are in the intersection of $\bar{W}_1$ and $\tilde{W}_1$, and $\bar{W}_2$ and $\tilde{W}_2$ respectively. In Step 2 assign the remaining d-polygons of $\bar{W}_1$, $\bar{W}_2$, $\tilde{W}_1$ and $\tilde{W}_2$. Finally, in Step 3 assign the d-polygons that belong to neither of $\bar{W}_1$, $\bar{W}_2$, $\tilde{W}_1$ and $\tilde{W}_2$.

Step 1: First assign a total value of $\min\{\bar{X}_1, \tilde{X}_1\}$ among the d-polygons that are in the intersection of $\bar{W}_1$ and $\tilde{W}_1$ by using Lemma 4.1. If $\bar{X}_1 \leq \tilde{X}_1$ (alternatively if $\bar{X}_1 > \tilde{X}_1$), then consider the d-lunes of $\bar{W}_1$ (alternatively $\tilde{W}_1$) as the rows of M

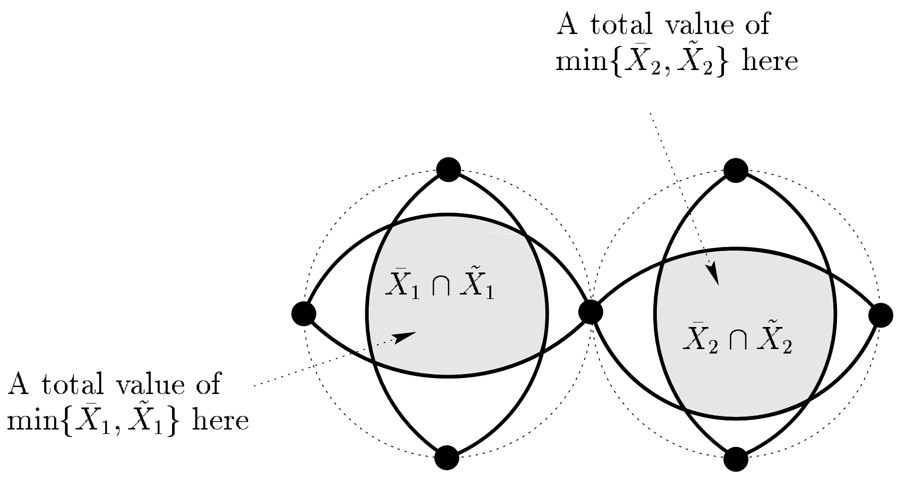

A total value of $\min\{\bar{X}_2, \tilde{X}_2\}$ here
$\bar{X}_1 \cap \tilde{X}_1$
$\bar{X}_2 \cap \tilde{X}_2$
A total value of $\min\{\bar{X}_1, \tilde{X}_1\}$ here

Figure 4.11: Illustrating Step 1 of Claim 4.2 by showing two opposite views of s.

and their final view differences as the elements of R, and similarly the d-lunes of $\tilde{W}_1$ (alternatively $\bar{W}_1$) as the columns of M and and their final view differences as the elements of C. After applying the lemma, the value of each cell of M becomes the value of the final view difference of the corresponding d-polygon. Similarly, assign a total value of $\min\{\bar{X}_2, \tilde{X}_2\}$ among the d-polygons that are in the intersection of $\bar{W}_2$ and $\tilde{W}_2$. See Figure 4.11

Step 2: This step will assign: (i) a total value of $\bar{X}_1 - \min\{\bar{X}_1, \tilde{X}_1\}$ among the remaining d-polygons of $\bar{W}_1$, which are in the intersection of $\bar{W}_1$ and $\tilde{\theta}_0$, or $\bar{W}_1$ and $\tilde{\theta}_{\tilde{K}}$, (ii) a total value of $\bar{X}_2 - \min\{\bar{X}_2, \tilde{X}_2\}$ among the remaining d-polygons of $\bar{W}_2$, which are in the intersection of $\bar{W}_2$ and $\tilde{\theta}_0$, or $\bar{W}_2$ and $\tilde{\theta}_{\tilde{K}}$, (iii) a total value of $\tilde{X}_1 - \min\{0, \tilde{X}_1, \bar{X}_1\}$ among the remaining d-polygons of $\tilde{W}_1$, which are in the intersection of $\tilde{W}_1$ and $\bar{\theta}_0$ or $\tilde{W}_1$ and $\bar{\theta}_{\bar{K}}$, and (iv) a total value of $\tilde{X}_2 - \min\{0, \tilde{X}_2, \bar{X}_2\}$ among the remaining d-polygons of $\tilde{W}_2$, which are in the intersection of $\tilde{W}_2$ and $\bar{\theta}_0$ or $\tilde{W}_2$ and $\bar{\theta}_{\bar{K}}$. The above four assignments will be done by using Lemma 4.1 twice, once for the first two and another for the remaining two. For the first case

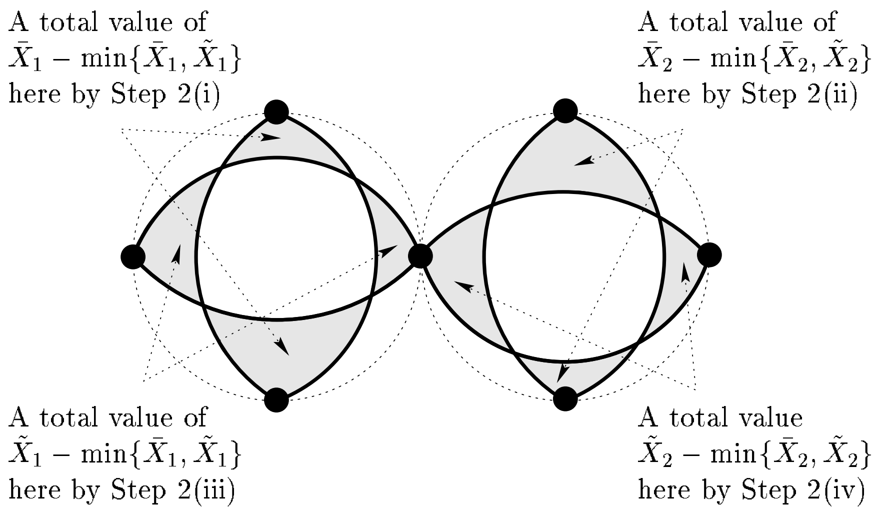

A total value of $\bar{X}_1 - \min\{\bar{X}_1, \tilde{X}_1\}$ here by Step 2(i)
A total value of $\bar{X}_2 - \min\{\bar{X}_2, \tilde{X}_2\}$ here by Step 2(ii)
A total value of $\tilde{X}_1 - \min\{\bar{X}_1, \tilde{X}_1\}$ here by Step 2(iii)
A total value $\tilde{X}_2 - \min\{\bar{X}_2, \tilde{X}_2\}$ here by Step 2(iv)

Figure 4.12: Illustrating Step 2 of Claim 4.2 by showing two opposite views of s.

the d-lunes of $\bar{W}_1$ and $\bar{W}_2$ correspond to the rows of M. Each element of R is the final view difference of the corresponding d-lune *minus* the total value already assigned to the d-polygons of that d-lune in the previous step (i.e., in Step 1). The d-lunes $\tilde{\theta}_0$ and $\tilde{\theta}_{\tilde{K}}$ correspond to the columns of M and their final view differences are the elements of C. Because of condition $(c1)$, the sum of all elements of R, which is $\bar{X}_1 - \min\{\bar{X}_1, \tilde{X}_1\} + \bar{X}_2 - \min\{\bar{X}_2, \tilde{X}_2\}$, is no more than the sum of all elements of C, which is $\tilde{\delta}_0 + \tilde{\delta}_{\tilde{K}} + (N - \tilde{D})$, which justifies the use of Lemma 4.1. After applying Lemma 4.1 the value of each element of M becomes the value of the final view difference of the corresponding d-polygon.

Perform the remaining two assignment similarly and use $(c2)$ as justification for using Lemma 4.1. This ends Step 2. (See Figure 4.12).

Step 3: After the above two steps, we have assigned a total value of $\max\{\bar{X}_1, \tilde{X}_1\}$ among the d-polygons of $\bar{W}_1$ and $\tilde{W}_1$ and a total value of $\max\{\bar{X}_2, \tilde{X}_2\}$ among the d-polygons of $\bar{W}_2$ and $\tilde{W}_2$. Those assignments, through Lemma 4.1, also guarantee

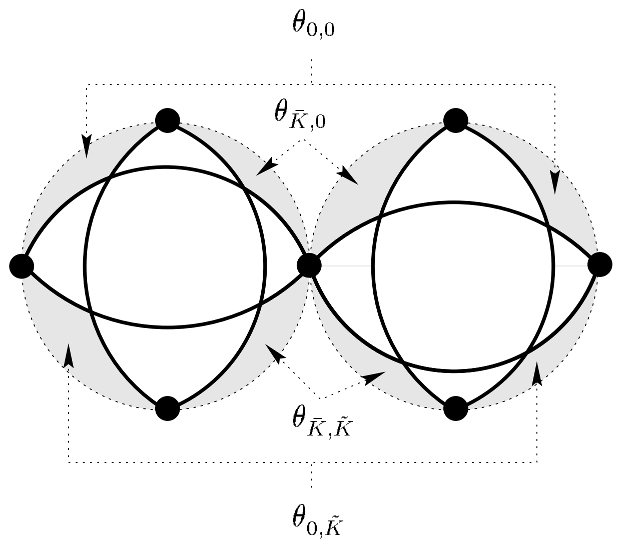


Figure 4.13: The d-polygons $\theta_{0,0}$, $\bar{\theta}_{\bar{K},0}$, $\tilde{\theta}_{0,\tilde{K}}$ and $\tilde{\theta}_{\bar{K},\tilde{K}}$ are assigned in Step 3 of Claim 4.2.

that for each of the d-lunes in $\bar{W}_1$, $\tilde{W}_1$, $\bar{W}_2$ and $\tilde{W}_2$, the poly-sum is equal to the final view difference. This last step will assign a total value of $N - \max\{\bar{X}_1, \tilde{X}_1\} - \max\{\bar{X}_2, \tilde{X}_2\}$ among the d-polygons that are not in any of $\bar{W}_1$, $\tilde{W}_1$, $\bar{W}_2$ and $\tilde{W}_2$. These d-polygons are $\theta_{0,0}$, $\bar{\theta}_{\bar{K},0}$, $\tilde{\theta}_{0,\tilde{K}}$ and $\tilde{\theta}_{\bar{K},\tilde{K}}$. See Figure 4.13. It will also ensure that when $\mathbf{max}\{\bar{D}, \tilde{D}\} < N$, there are two opposite d-polygons with positive final view differences.

Remember that we assumed $\bar{D} < \tilde{D}$ when $\bar{D} \neq \tilde{D}$. Observe that when $\tilde{D} < N$, from the assignment of final view differences of the d-lunes, all of $\bar{\Delta}_0$, $\bar{\Delta}_{\bar{K}}$, $\tilde{\Delta}_0$ and $\tilde{\Delta}_{\tilde{K}}$ are positive. It also implies that $N - \bar{X}_1 - \bar{X}_2 \geq 1$, and $N - \tilde{X}_1 - \tilde{X}_2 \geq 1$. Choose the d-i pair of opposite d-polygons $\theta_{0,0}$ and $\theta_{\bar{K},\tilde{K}}$ (the d-i pair of $\theta_{0,\tilde{K}}$ and $\theta_{\bar{K},0}$ would also work) and set each of their final view differences as one. This two opposite d-polygons prove the second part of the original lemma.

Then assign the remaining value of $N - \max\{\bar{X}_1, \tilde{X}_1\} - \max\{\bar{X}_2, \tilde{X}_2\}$ or $N - \max\{\bar{X}_1, \tilde{X}_1\} - \max\{\bar{X}_2, \tilde{X}_2\} - 2$, as appropriate after the previous paragraph, among $\theta_{0,0}$,$\theta_{\bar{K},0}$, $\theta_{0,\tilde{K}}$ and $\theta_{\bar{K},\tilde{K}}$ by applying Lemma 4.1 as follows. Here M is a 2×2 matrix with $\bar{\theta}_0$ and $\bar{\theta}_{\bar{K}}$ corresponding to two rows and $\tilde{\theta}_0$ and $\tilde{\theta}_{\tilde{K}}$ corresponding to two columns. One element of R is $\bar{\Delta}_0$ minus the total value already assigned to the d-polygons of $\bar{\theta}_0$ and the other element is $\bar{\Delta}_{\bar{K}}$ minus the total value already assigned to the d-polygons of $\bar{\theta}_{\bar{K}}$. Similarly, one element of C is $\tilde{\Delta}_0$ minus the total value already assigned to the d-polygons of $\tilde{\theta}_0$ and the other element is $\bar{\Delta}_{\tilde{K}}$ minus the total value already assigned to the d-polygons of $\tilde{\theta}_{\tilde{K}}$. After applying the lemma the value of each of $\bar{\theta}_0$, $\bar{\theta}_{\bar{K}}$, $\tilde{\theta}_0$ and $\tilde{\theta}_{\tilde{K}}$ is the value of the corresponding element of M plus its previous value, which is zero or one, as appropriate, from the previous paragraph. This completes Step 3 and the assignment.

For justification, Lemma 4.1 guarantees that for any d-lune θ of $\bar{\mathcal{S}}$ or $\tilde{\mathcal{S}}$ the poly-sum is equal to the final view difference of θ. So for any d-i pair $\langle \bar{d}_i, \bar{n}_i \rangle$ of $\bar{\mathcal{S}}$, the poly-sum of $\bar{h}_i$ is $\sum_{l=i+\bar{K}}^{i-1} \bar{\delta}_l + \frac{1}{2}(N - \bar{D})$, which is $\bar{n}_i$ according to Lemma 3.2. The justification for a d-i pair in $\tilde{\mathcal{S}}$ is similar. □

With the above claim in hand one can decide whether a valid assignment is possible or not by checking the conditions $(c1)$ and $(c2)$ and then can find an assignment, when it exists, as in the manner indicated in the proof of the claim.

We now study the time complexity. Computing $\bar{W}_1$, $\bar{W}_2$, $\tilde{W}_1$, $\tilde{W}_2$, $\bar{X}_1$, $\bar{X}_2$, $\tilde{X}_1$ and $\tilde{X}_2$ takes a total $O(\bar{K} + \tilde{K})$ time. Checking $(c1)$ and $(c2)$ takes constant time. The matrix size in Step 1 is $(\bar{K}-1)\times(\tilde{K}-1)$ for both cases. So Lemma 4.1 in Step 1 takes $O(\bar{K} + \tilde{K})$ time. The size of the two matrices in Step 2 are $(2(\bar{K} - 1) \times 2)$ and $(2(\tilde{K} - 1) \times 2)$ respectively. So Step 2 takes $O(\bar{K} + \tilde{K})$ time too. Finally, the matrix size in Step 3 is (2×2) and thus Step 3 takes constant time. So the overall time is $O(\bar{K} + \tilde{K})$. $\square$

4.4.6 Valid selection for arbitrary visibility

After obtaining a valid assignment, we will find a valid selection of N normal-points on s. Remember that when the directions see arbitrary number of faces, we can not find a valid selection for all cases. The condition when we can find is $\mathbf{max}\{\bar{D}, \tilde{D}\} < N$. So throughout this subsection assume that $\mathbf{max}\{\bar{D}, \tilde{D}\} < N$.

The overall idea of choosing N normal-points is as follow: Select $N-2$ normal-points arbotrarily (according to $\Delta_{i,j}$), and the remaining two normal-points in two opposite d-polygons such that they preclude any hemisphere from containing all. The flowing auxiliary lemma will be used to select the last two normal-points.

An auxiliary lemma

Lemma 4.4 *Let x_1, x_2 and x_3 be three points of s such that they are not all on a great circle. Let t be the (spherical) triangle defined by the segments $\overline{x_1x_2}$, $\overline{x_2x_3}$ and $\overline{x_3x_1}$. Let t' be the opposite of t. Let x_4 be a point strictly within t'. Then, there does not exists a hemisphere of s containing all of x_1, x_2, x_3 and x_4.*

Proof. Let h be a hemisphere of s containing x_1, x_2, x_3. Then the interior of t' must be strictly within $s \backslash h$, which implies that x_4 is strictly within $s \backslash h$. So h does not contain x_4. $\square$

The valid selection

Lemma 4.5 *Given $\bar{\mathcal{S}}$ and $\tilde{\mathcal{S}}$, and a valid assignment of total value of N among the d-polygons of s, if $\mathbf{max}\{\bar{D}, \tilde{D}\} < N$, then one can select in $O(N + \bar{K} + \tilde{K})$ time N normal-points from s such that for each d-polygon $\theta_{i,j}$ the number of normal-points strictly within it is $\Delta_{i,j}$ and not all N normal-points intersect a hemisphere of s.*

Proof. From Lemma 4.2 and Lemma 4.3 remember that when $\mathbf{max}\{\tilde{D}, \tilde{D}\} < N$, there are two positive opposite d-polygons $\theta_{0,0}$ and $\theta_{\bar{K},\tilde{K}}$. Choose all normal-points arbitrarily except one from each of $\theta_{0,0}$ and $\theta_{\bar{K},\tilde{K}}$. Let x_1 and x_2 be two of the chosen $N - 2$ normal-points. If (by any chance) x_1 and x_2 are antipodal, then move one slightly strictly within its respective d-polygon. Now consider a hemisphere h that contains x_1 and x_2 strictly within it; this exists since x_1 and x_2 are not antipodal.

For rest of the proof please refer to Figure 4.14. At least one of $\theta_{0,0}$ and $\theta_{\bar{K},\tilde{K}}$ intersects h. Say $r = \theta_{0,0} \cap h \neq \emptyset$. Add one normal-point $x_3 \neq x_1, x_2$ strictly within r. such that x_1, x_2, x_3 do not lie on a great circle. (Observe that since $\theta_{0,0}$ is strictly within a hemisphere, r can be at most h and thus it is possible that x_1, x_2, x_3 lie on a great circle). For the final normal-point, consider the spherical triangle defined by the three segments $\overline{x_1x_2}$, $\overline{x_2x_3}$ and $\overline{x_3x_1}$. Since x_3 is strictly within r, this triangle intersects r. Let this intersection be t. Let the opposite of t be t', which is a subset of $\theta_{\bar{K},\tilde{K}}$. The last normal-point is an arbitrary point x_4 strictly within t'. From Lemma 4.4, x_1, x_2, x_3 and x_4 do not intersect a hemisphere of s.

Now consider the time complexity. From any d-polygon one can choose any number of normal-points strictly within it in linear time. Moreover, while using Lemma 4.1 to find a valid assignment in Lemma 4.2 and Lemma 4.3, one can get the list of all positive d-polygons in total of $O(\bar{K} + \tilde{K})$ time. So the selection of

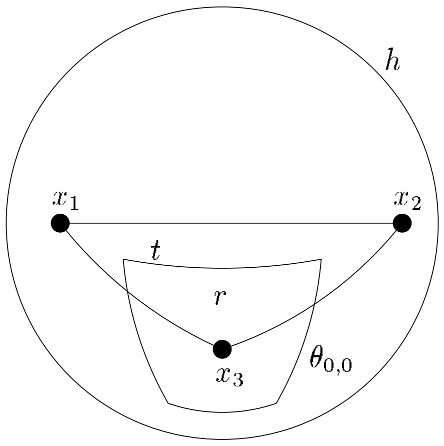


Figure 4.14: Illustrating the proof of Lemma 4.5.

first $N-2$ points from the positive d-polygons can be done in $O(N+\bar{K}+\tilde{K})$ time. Selecting x_1 and x_2 from already chosen normal-points and shifting one of them, if necessary, takes constant time. Computing h should take constant time. As the size of each d-polygon (in terms of segments) is four, computing the intersection of $\theta_{0,0}$ and $\theta_{\bar{K},\tilde{K}}$ with h takes constant time. For similar reasons, computing t and t' also takes constant time. Finally, choosing x_3 and x_4 takes constant time. So the overall time is $O(N+\bar{K}+\tilde{K})$. □

4.4.7 Valid selection for limited visibilty

In this section we choose normal-points when all direction see at least four faces. We show that there always exists a valid selection and show how to find one. The overall idea of the selection is as follows. Any three great circles without two antipodal points common to all of them divide s into eight octants. As a preliminary, we will first prove (in the following subsection) that any eight points strictly within these eight octants, respectively, can not intersect a single hemisphere. Then we will prove that when all directions see at least four faces, there always exist three such great circles so that it is possible to find a valid selection with at least one normal-point in each of the eight octants (This will be proven without finding the actual valid selection). Then we will first find three such great circles. Finally we select all N normal-points as follows. Initially we will select them arbitrarily according to $\Delta_{i,j}$. This initial selection may not gurantee that all eight octants get at least one normal-points each. If such case happens, we will move some normal-points so that no octant remain empty. This movement may change the valid assignment (remember that there may be more than one valid assignment.)

An auxiliary lemma

Lemma 4.6 *Consider any three great circles of* s *that do not intersect in a common d-i pair of antipodal points. The arrangement of these three great circles divides* s *into eight octants. For each octant consider an arbitrary point that is strictly within it. Then, these eight points cannot intersect a hemisphere of* s*.*

Proof. This proof is similar to the proof of Claim 4.1 in Section 4.3.

Let g_1, g_2 and g_3 be the three great circles and let u_1, u_2 and u_3 be the normals of the planes that define g_1, g_2 and g_3. Pick eight arbitrary points that are strictly within the eight octants of g_1, g_2 and g_3. By way of contradiction assume that there exists a hemisphere h that intersects all of them. Let u be the normal of the plane of the great circle of h and let u be in h. Since g_1, g_2 and g_3 do not have a common d-i pair of antipodal points, their normals form a basis of 3D space, and hence one can write: $u = \lambda_1 u_1 + \lambda_2 u_2 + \lambda_3 u_3$ for some value of λ_1, λ_2 and λ_3. Note that λ_1, λ_2 and λ_3 cannot all be zero since u is non-zero.

Now pick an octant as follows. If $\lambda_1 \geq 0$, pick the hemisphere of g_1 that contains n_1; else pick the other hemisphere. Similarly pick a hemisphere from g_2 and g_3. Take the intersection of these three hemispheres. Let the resulting octant be o^+, and let p^+ be the chosen point in it. Since p^+ is strictly within o^+, $\langle \overrightarrow{p^+} \cdot u_i \rangle \neq 0$[1], for $i = 1, 2, 3$. By our choice of octants, we also have the same *sign* for λ_i and $\langle \overrightarrow{p^+} \cdot u_i \rangle$ for those $i = 1, 2, 3$ for which $\lambda_i \neq 0$. Therefore $\langle \overrightarrow{p^+} \cdot u \rangle = \sum_{i=1}^{3} \lambda_i \langle \overrightarrow{p^+} \cdot u_i \rangle \geq 0$, and in fact we have $\langle \overrightarrow{p^+} \cdot u \rangle > 0$ since not all λ_i can be zero.

Now let o^- be the octant opposite to octant o^+, and let p^- be the chosen point in it. Since p^- is strictly within o^-, $\langle \overrightarrow{p^-} \cdot u_i \rangle \leq 0$. Here the signs of λ_i and $\langle \overrightarrow{p}^- \cdot u_i \rangle$ are opposite for those $i = 1, 2, 3$ for which $\lambda_i \neq 0$. Therefore $\langle \overrightarrow{p^-} \cdot u \rangle = \sum_{i=1}^{3} \lambda_i \langle \overrightarrow{p^-} \cdot u_i \rangle \leq 0$, and in fact we have $\langle \overrightarrow{p^-} \cdot u \rangle < 0$ since not all λ_i can be zero. Thus p^+ and p^- are on opposite hemispheres defined by u, and not all eight points intersect the hemisphere h. $\square$

Valid selection

Instead of directly finding three great circles for the whole set $\mathcal{S}$, first we will select a great circle g_{ab} common for $\bar{\mathcal{S}}$ and $\tilde{\mathcal{S}}$ and then find a great circle g_1 for $\bar{\mathcal{S}}$ and another great circle g_b for $\tilde{\mathcal{S}}$. We will find g_{ab}, g_a, g_b in such a way that each of the four lunes created by g_{ab} and g_a, and g_{ab} and g_b, can have at least two normal points each. It will allow us to select N normal-points so that the eight octants created by these two sets of four lunes contain at least one point each.

All of these are done in the following two lemmas. In the first lemma we will show that for any proper d-i set with all its integers at least four and all directions planar (i.e. a d-i set like $\bar{\mathcal{S}}$ or $\tilde{\mathcal{S}}$), given one great circle we can always find another

[1]$\langle x \cdot y \rangle$ means the inner product of the vector x and y.

great circle so that four lunes created by them can contain at least two points each. In the second lemma we will select the actual position of the normal-points.

Lemma 4.7 *Given a proper d-i set $\mathcal{S}$ of size $2K$, where the directions are planar and $n_i \geq 4$ for any d-i pair $\langle d_i, n_i \rangle$, assume that we have assigned the final view differences of the d-lunes of $\mathcal{S}$ such that for any i, $n_i = \sum_{j=i-K}^{i-1} \Delta_j$. Then, given an arbitrary great circle g_1 passing through the poles, it is always possible in $O(N)$ time to find another great circle g_2 also passing through the poles such that there exist a selection of N normal-points on s where for each d-i pair $\langle d_i, n_i \rangle$ of $\mathcal{S}$ the number of normal-points in the visible hemisphere h_i of d_i and each of the four lunes created by g_1 and g_2 contains at least two normal-points strictly within it.*

Proof. Let the four lunes that will be created by g_1 and g_2 be q_1, q_2, q_3 and q_4, where q_1, q_2, q_3, q_4 is their circular sequence. Maintain four variables x_1, x_2, x_3, and x_4 corresponding to the number of normal-points in q_1, q_2, q_3 and q_4 respectively. We will describe the algorithm in two steps: (1) Initialization and (2) Rotation.

Step 1: Initialization. In this step initialize g_2, x_1, x_2, x_3 and x_4. There are two cases depending on whether g_1 is a great circle of a visible hemisphere of some direction of $\mathcal{S}$ or not. First consider the case when g_1 is such a great circle. Without loss of generality let g_1 be the great circle of the visible hemisphere h_0 (and h_K). In this case assign g_2 to be same as g_1, q_1 and q_3 as the two quadrants whose widths are zero, q_2 and q_4 as h_0 and h_K respectively, and x_1 and x_3 as zero. (See Figure 4.15(a)). Note that with this assignment, all d-lunes are a subset of either q_2 or q_4. Assign x_2 as n_0 and x_4 as n_K.

Next consider the case when g_1 is not a great circle of a visible hemisphere of some direction of $\mathcal{S}$. Here g_1 intersects two opposite d-lunes and without loss of

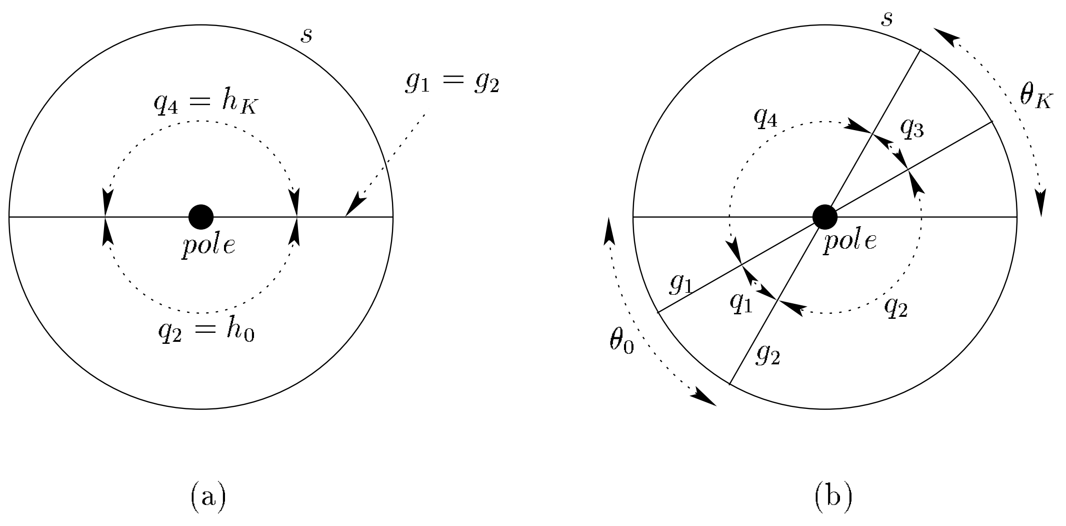


Figure 4.15: Initialization step of Lemma 4.7. Example of the case (a) when g_1 is a great circle of a visible hemisphere of $\mathcal{S}$, and (b) when it isn't.

generality let them be θ_0 and θ_K. Assign g_2 as the great circle of h_1, q_1 and q_3 as the two lunes that are subsets of θ_0 and θ_K respectively, q_2 as $h_1 \backslash q_3$, and q_4 as the opposite of q_3. (See Figure 4.15(b)). As before assign x_1 and x_3 as zero, x_2 as n_0, and x_4 as n_K. This concludes Step 1.

Step 2: Rotation. Rotate g_2 until we can assign one of x_1 and x_2 two and the other one at least two. When the rotation stops it is guaranteed that all of x_1, x_2, x_3 and x_4 are at least two— which will be proven in the justification. The rotation of g_2 is performed consistently in the direction which is used to define the order of the directions of $\mathcal{S}$. At any time θ_i and θ_{i+K} are the two opposite d-lunes from q_2 and q_4, respectively, such that they either intersect g_2 or are next to g_2. Initially $i = 0$ (from the Step 1 above).

At each step of rotation check the values of $x_1 + \Delta_i$ and $x_3 + \Delta_{i+K}$. If at least one of them is less than two, then do the followings: both increase x_1 and

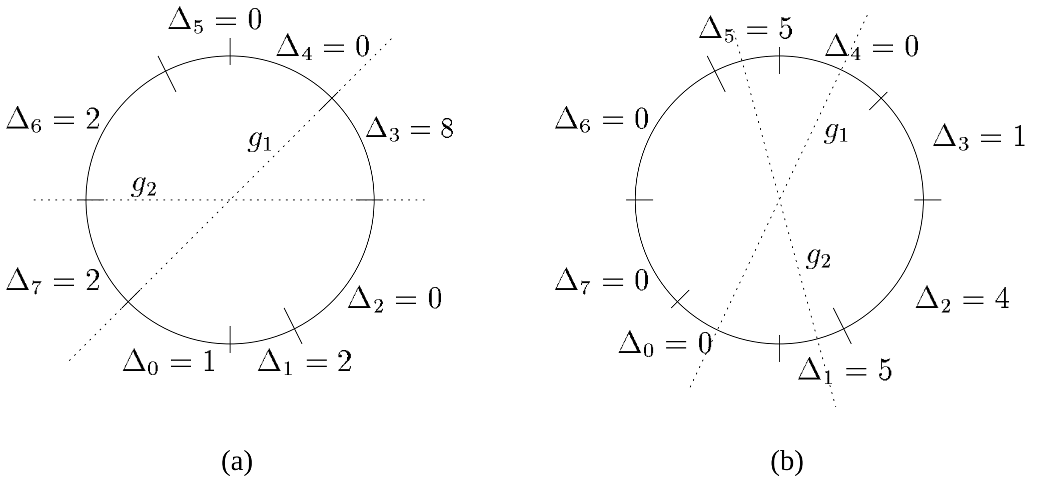


Figure 4.16: Example Two examples of selection of g_1 and g_2.

decrease x_2 by Δ_i, both increase x_3 and decrease x_4 by Δ_{i+K}, move g_2 to the great circle of h_{i+1}, increase i by one, and repeat. If both of them are at least two, do the followings: both increase x_1 and decrease x_2 by $\mathbf{max}\{0, 2 - x_1\}$, both increase x_3 and decrease x_4 by $\mathbf{max}\{0, 2-x_3\}$, move g_2 to the somewhere in between its current position and h_{i+1} or g_1, whichever comes first, and stop the rotation. This ends the algorithm for rotation.

Let us see some examples. Please refer to Figure 4.16, where the d-lunes are shown in 2D. In the first example, given g_1 as the great circle of d_0 and d_4, then the final position of g_2 is the great circle of d_2 and d_6. Here x_1, x_2, x_3, x_4 are three, eight, two, and two respectively. In the second example, given g_1, which intersects θ_0 and θ_4, the final position of g_2 intersects θ_1 and θ_5. Here x_1, x_2, x_3, x_4 are two, eight, two, three respectively.

Now we justify why the x_1, x_2, x_3 and x_4 normal-points can be selected from q_1, q_2, q_3, and q_4 respectively. We only show this for q_1; the others are similar. All x_1 normal-points are selected from the d-lunes of $\mathcal{S}$ that are strictly within q_1 or partially intersect q_1. (In fact, all d-lunes that intersect q_1 are strictly within q_1, except the "last" one which may intersect q_1 partially). During the initialization and rotation steps, one can keep track of the number of points that each d-lune "contributes" to this set of x_1 points. In fact, the number of points contributed by each d-lune which is strictly within q_1 is the final view difference. On the other hand, the number of points contributed by the partially intersected d-lune is one or two. One can select these x_1 normal-points from the intersection of these d-lunes and q_1 by avoiding their boundaries so that each d-lune gets the number of points equal to their contribution in x_1. This completes the description of the algorithm.

Justification. In the algorithm it was ensured that one among x_1 and x_3 be two and the other one be at least two. We now show that x_2 and x_4 are also at least two. The initial position of g_2 is the great circle of h_0 or a great circle intersecting the d-lunes θ_0 and θ_K. Let the final position of g_2 be the great circle of h_m or be a great circle intersecting θ_m and θ_{m+K} for some $0 \leq m \leq K-1$.

We first prove that $x_2 \geq 2$. The final value of x_2 is its initial value, which is n_0, minus the final value of x_1, which is two. But from the assumption, $n_0 \geq 4$. So $x_2 \geq 2$.

Next we prove that $x_4 \geq 2$. Let $s_1 = \sum_{i=1}^{m-1} \Delta_i$. From the algorithm, x_1 can be written as $s_1 + \mathbf{max}\{0, 2 - s_1\}$, where the term $\mathbf{max}\{0, 2 - s_1\}$ is the updated value of x_1 in the last step. As $x_1 = 2$, $s_1 \leq 2$. Similarly, let $s_2 = \sum_{i=m+K}^{0} \Delta_i$. From the algorithm, x_4 can be written as $s_2 + \mathbf{max}\{0, 2 - s_2\}$. So $x_4 \geq s_2$. Now consider n_m, which, from the given conditions, is at least four. Moreover, from the given condition, n_m is $\sum_{j=m-K}^{m-1} \Delta_j$, which in turn can be written as $s_1 + s_2$. So $s_1 + s_2 \geq 4$. We already know that $s_1 \leq 2$. Using this inequality in $s_1 + s_2 \geq 4$, we get $s_2 \geq 2$. Since $x_4 \geq s_2$, we get $x_4 \geq 2$.

Finally, consider the time complexity. In the initialization step, finding the position of g_1 with respect to the great circles of the visible hemispheres of $\mathcal{S}$ takes $O(K)$ time. All other initialization take constant time. Now, for the number of moves for g_2 during the rotation, observe that before g_2 moves through the whole hemisphere of h_0 and the whole hemisphere of h_K, both x_1 and x_3 become at least two since n_0 and n_K are at least four. So the number of moves by g_2 is at most K.

During the rotation, each step of the rotation takes constant time. So total time for rotation is $O(K)$. The overall time is thus $O(K)$. $\square$

We can now proceed with the actual selection of N normal-points from s.

Lemma 4.8 *Given $\bar{\mathcal{S}}$ and $\tilde{\mathcal{S}}$, where $n \geq 4$ for each d-i pair $\langle d, n \rangle$ in $\bar{\mathcal{S}}$ or $\tilde{\mathcal{S}}$, and given a valid assignment of a total value of N among the d-polygons, then one can select in $O(N + \bar{K} + \tilde{K})$ time N normal-points on s such that for each d-polygon the number of normal-points is equal to its final view difference and all N normal-points do not intersect a hemisphere of s.*

Proof. Remember that after the valid assignment for each d-lune of $\bar{\mathcal{S}}$ or $\tilde{\mathcal{S}}$ the final view difference and the poly-sum are the same. That means one can write, for any i, $\bar{n}_i = \sum_{m=i-\bar{K}}^{i-1} \bar{\Delta}_i$, and for any j, $\tilde{n}_i = \sum_{m=i-\tilde{K}}^{i-1} \tilde{\Delta}_i$. Now, let g_{ab} be the great circle that passes through the poles of $\bar{\mathcal{S}}$ and $\tilde{\mathcal{S}}$. Apply Lemma 4.7 twice, once for $\bar{\mathcal{S}}$ and once for $\tilde{\mathcal{S}}$, by considering g_{ab} as the given great circle in both cases. Let the two great circles that we get from Lemma 4.7 for $\bar{\mathcal{S}}$ and $\tilde{\mathcal{S}}$ be g_a and g_b respectively. The great circles g_{ab}, g_a and g_b divide s into eight octants. We will only show how to select normal-points from one hemisphere of g_{ab}. Selecting normal-points from the other hemisphere is similar. Any normal-point will avoid the boundary of the d-polygons.

Let o_1, o_2, o_3 and o_4 be the four octants of the hemisphere of g_{ab} that are being considered, and assume that o_1 and o_2 are in one hemisphere of g_a and o_2 and o_3 are in one hemisphere of g_b. (See Figure 4.17(a)). The selection process have two steps.

Step 1: First find all positive d-polygons that are strictly within o_1, o_2, o_3 or o_4. For each such positive d-polygon, select all its points strictly within it. Then look at the positive d-polygons that intersect two or all of o_1, o_2, o_3 and o_4. (Observe that a d-polygon cannot intersect three octants and there can be at most one d-polygon that intersects all of them). This type of octant may arise when g_a or g_b or both are not the great circles of visible hemispheres of $\bar{\mathcal{S}}$ and $\tilde{\mathcal{S}}$. To select their normal-points look at the intersecting octants. If there are octants that still do not have any normal-point selected, select normal-points in such a way that these octants get at least one normal-point each.

Step 2: After selecting normal-points in Step 1 it may still be possible that there are some octants that do not have any normal-point selected. This may happen when an octant does not intersect any positive d-lune at all, and those octants are called *empty*. The following claim will show that if there exists any empty octant, then one can *re*assign d-polygons and *re*select normal-points among them such that there does not remain any empty octant.

Claim 4.3 *Assume there are some empty octants. Then,*

- *The number of empty octants is at most two. Moreover, either o_1 and o_3 or o_2 and o_4 are non-empty.*
- *It is possible to reassign the final view differences of the d-polygons and reselect some normal-points among them so that each octant gets at least on normal-point and the validity of the assignment is preserved.*

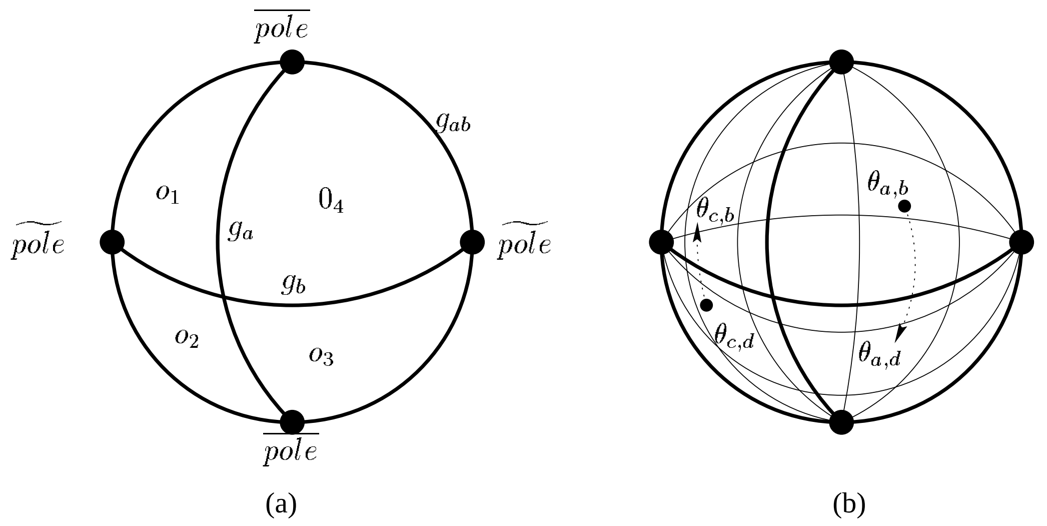


Figure 4.17: (a) The four octants o_1, o_2, o_3 and o_4. (b) Reassigning the d-polygons and reselecting the normal-points for the case of one or more empty d-polygons.

Proof. (i) By way of contradiction assume that there is only one positive d-polygon. Without loss of generality let it be o_1. This means that under any selection process the number of normal-points that can be selected from each of the pairs o_2 and o_3, and o_3 and o_4 is zero. But according to Lemma 4.7 it is possible to have a selection so that each of them get at least two normal-points. A contradiction. So there are at least two octants which intersect positive d-polygons. Moreover, by the same reason any two consecutive octants cannot be empty. So either o_1 and o_3, or o_2 and o_4 are not empty.

(ii) Without loss of generality assume that o_1 is an empty octant. By Lemma 4.7 as o_1 and o_2 together have at least two normal-points, o_2 intersects positive d-polygons with their total view differences at least two. Similarly, o_4 intersects positive d-polygons with their total view differences at least two. Let $\theta_{a,b}$ and $\theta_{c,d}$ be two positive d-polygons intersecting o_2 and o_4 respectively. Recall that $\theta_{a,b} = \bar{\theta}_a \cap \tilde{\theta}_b$ and $\theta_{c,d} = \bar{\theta}_c \cap \tilde{\theta}_d$. (See Figure 4.17(b)). Observe that the intersection of $\bar{\theta}_a$ and $\tilde{\theta}_d$, which is the d-polygon $\theta_{a,d}$, intersects o_1. Similarly, the intersection of $\bar{\theta}_c$ and $\tilde{\theta}_b$, which is the d-polygon $\theta_{c,b}$, intersect o_3.

Now we perform the reassignment and reselection. For reselection, remove one normal-point from each of $\theta_{a,b}$ and $\theta_{c,d}$ and select one new normal-point from each of $\theta_{a,d}$ and $\theta_{c,b}$. For reassignment, decrease the value of $\Delta_{a,b}$ and $\Delta_{c,d}$ by one and increase the value of $\Delta_{a,d}$ and $\Delta_{c,b}$ by one. Figure 4.17(b) shows the reassignment and reselection as one normal-point moving from $\theta_{a,b}$ to $\theta_{a,d}$ and another normal-point moving from $\theta_{c,d}$ to $\theta_{c,b}$.

The justification is easy to follow. The view difference for any d-lunes has not been changed. Moreover, after Step 1 both o_1 and o_3 had at least two normal-points each and after the reselection in Step 2 they still have at least one normal-point each. So the modified assignment of the d-polygons is valid and after the modified selection all octants get at least one point each. $\square$

With the end of this claim, Step 2 as well as the whole selection process is completed.

We now study the time complexity of the entire selection process. We consider the time required to select normal-points from o_1, o_2, o_3 and o_4 only, as the time required to select normal-points from other three octants is the same. As mentioned in the proof of Lemma 4.5, while using Lemma 4.1 to find a valid assignment in Lemma 4.2 and Lemma 4.3, one can get the list of all positive d-polygons in total of $O(\bar{K}+\tilde{K})$ time, and within the same order of time one can find their intersection with the octants.

Normal-points within all the positive d-polygons that intersect one octant each can be selected in linear time. A d-polygon can intersect at most four octants. So for each d-polygon that intersects more than one octant the normal-points can be selected from the intersecting octants with a target of minimizing the number of empty octants, which again can be done in linear time. So the Step 1 takes $O(N)$ time. Finally, Claim 4.3 and thus Step 2 take constant time. So overall time is $O(N + \bar{K} + \tilde{K})$. $\square$

The following theorem summerizes the results.

Theorem 4.2 *Given a proper d-i set $\mathcal{S}$ and an integer $N \geq 4$, where the directions of $\mathcal{S}$ are in two planes through the origin, then one can construct a feasible polyhedron P, if one exists, in $O(N \log N + \bar{K} + \tilde{K})$ time, where $\bar{K}$ and $\tilde{K}$ are the number of directions in the two planes, for the following two cases:*

- *$n \geq 4$ for each d-i pair $\langle d, n \rangle$ in $\mathcal{S}$, and*
- *$\max\{\bar{D}, \tilde{D}\} < N$.*

Corollary 4.2 *If the feasible polyhedron is allowed to be unbounded, it can be always constructed, when it exists, in $O(N \log N + \bar{K} + \tilde{K})$ time.*

4.4.8 A valid assignment is not enough

This section shows by an example why the technique of finding a valid assignment and a then valid selection fails to decide the existence of a feasible polyhedron for $\mathcal{S}$ when $n \leq 3$ for some d-i pair $\langle d, n \rangle$ of $\mathcal{S}$. The idea is to show that for any valid assignment there may not be a valid selection.

We refer to Figure 4.18. First consider the proper d-i set of (a). It has twelve d-i pairs and $N = 4$. The only positive view differences are δ_0, δ_4, and δ_8 with respective values value one, one, and two respectively. So $D = N$, and by Corollary 4.1 there always exists a feasible polyhedron for this proper d-i set. The key property of this proper d-i set is that the two directions in each d-i pair of (d_0, d_1), (d_4, d_5), and (d_8, d_9) are almost parallel and for that reason their corresponding positive d-lunes on s are very thin. Moreover, the circular distance between other d-i pairs of consecutive directions can be adjusted to increase/decrease the relative circular distance among the positive d-lunes. For example, (b) and (c) show the positive d-lunes for two different versions $\bar{\mathcal{S}}$ and $\tilde{\mathcal{S}}$ of the proper d-i set of (a).

Now, consider a proper d-i set $\mathcal{S}$ which is the union of $\bar{\mathcal{S}}$ and $\tilde{\mathcal{S}}$. There are two possible valid assignments for the d-polygons of $\mathcal{S}$ which are shown in (c).

Let us explain these two assignments. Consider $\bar{\mathcal{S}}$. As $\bar{D} = N$, Lemma 3.2 implies that for any d-i pair $\langle \bar{d}_i, \bar{n} \rangle$ in $\bar{\mathcal{S}}$, in any valid assignment the poly-sum of h_i should be $\sum_{i=0}^{\bar{K}} \bar{\delta}_i$, which further implies that for each d-lune $\bar{\theta}_i$ of $\bar{\mathcal{S}}$ the poly-sum should be $\bar{\delta}_i$. Similarly for each d-lune $\tilde{\theta}_j$ of $\tilde{\mathcal{S}}$ the poly-sum should be $\tilde{\delta}_j$. So in the example the poly-sum of each of $\bar{\theta}_0$, $\bar{\theta}_4$, $\tilde{\theta}_0$, and $\tilde{\theta}_4$ should be one and the poly-sum of each of $\bar{\theta}_8$ and $\tilde{\theta}_8$ should be two. That means the d-polygon $\bar{\theta}_{8,8}$ must be assigned two, and among the d-polygons $\bar{\theta}_{0,0}$, $\bar{\theta}_{0,4}$, $\bar{\theta}_{4,0}$, and $\bar{\theta}_{4,4}$ either $\bar{\theta}_{0,0}$ and $\bar{\theta}_{4,4}$ or $\bar{\theta}_{0,4}$ and $\bar{\theta}_{4,0}$ should be assigned one. No other d-polygon can be positive. But in either case all three positive d-polygons are strictly within a single hemisphere as shown in (d). So no valid selection can exist.

4.5 Non-convex polyhedra

This section briefly studies the problem of creating non-convex polyhedra from orthogonal projections in the case when at least four faces are visible from every direction.

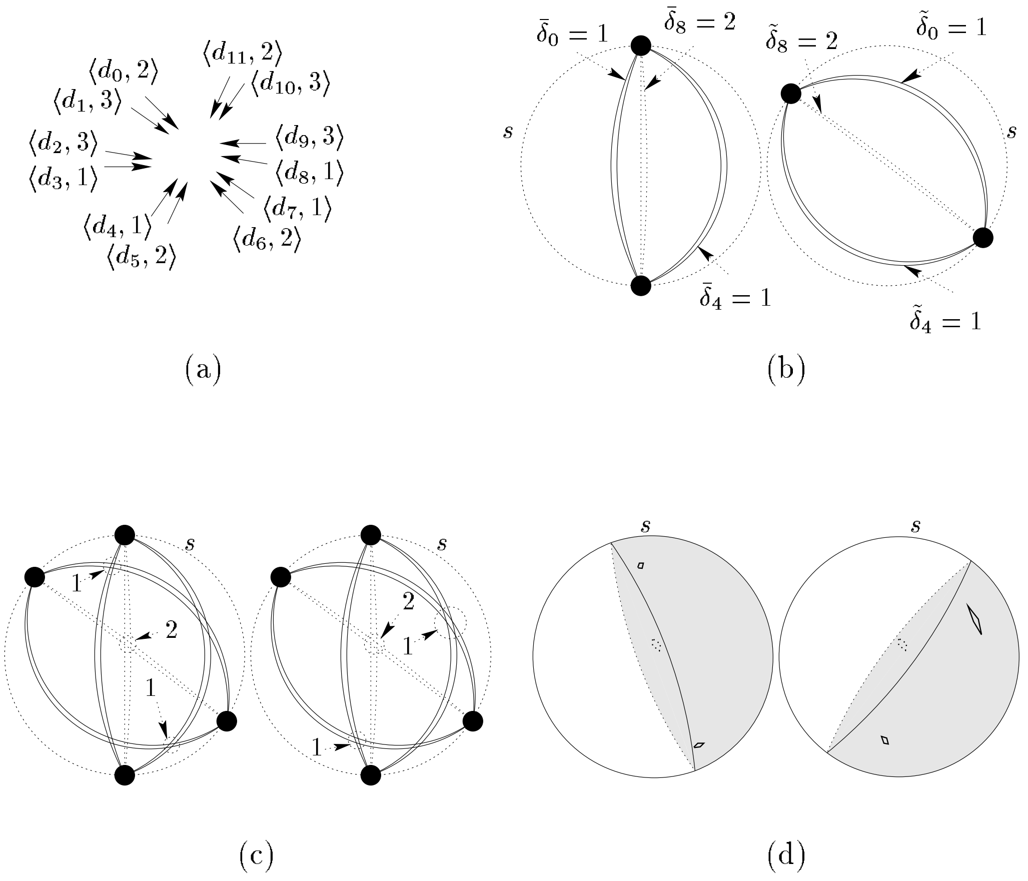


Figure 4.18: (a) A sample proper d-i set from which will give two proper d-i set $\bar{\mathcal{S}}$ and $\tilde{\mathcal{S}}$ for the example. (b) Positive d-i pairs for $\bar{\mathcal{S}}$ and $\tilde{\mathcal{S}}$. Here each of $\bar{\mathcal{S}}$ and $\tilde{\mathcal{S}}$ is similar to the proper d-i set in (a) except that the relative distance among the directions has been adjusted. (c) The two possible valid assignments. (d) For each valid assignment all positive d-polygons are strictly within a hemisphere of s.

4.5.1 The special case

The differences between creating convex and non-convex polyhedron are similar to those for convex and non-convex polygons mentioned in Chapter 3. are quickly reviewed here. For a non-convex polyhedron, the visibility of a face from a view direction may be partial or full, whereas for a convex polyhedron a face is always fully visible. A face of a non-convex polyhedron may be parallel to a view direction while it is not visible at all; so it is not clear whether a projection is to be considered as degenerate when this type of degeneracies happen. On the other hand, for a convex polyhedron, a degenerate face is visible as a line segment. Finally, the size of a non-convex polyhedron does not depend upon the number visible faces from the view directions, whereas two opposite non-degenerate projections uniquely determine the size of a convex polyhedron.

Keeping the above difficulties in mind, we will study a special case with the assumption that the size of a *feasible polyhedron* is not given, the partially visible edges are not counted as visible, and the number of visible faces from each direction is at least four The size of this feasible polyhedron will be linear to the sum of the number of visible faces from all directions. Reconstruction under alternative definitions of visibility, degeneracy of a face, and/or the size of the feasible polyhedron are left as open problems. (The coputation model is assumed to be a ral-RAM.)

4.5.2 The construction

Theorem 4.3 *Let S be a d-i set of size K. For any d-i pair $\langle d, n\rangle$ in S, let $n \geq 4$. Let N be the sum of the integers in S. Then it is always possible to create a non-convex polyhedron P of size $O(N)$ such that for each d-i pair $\langle d, n\rangle$ in S the faces of P that are not parallel to d and are visible from d is n. Moreover, the time*

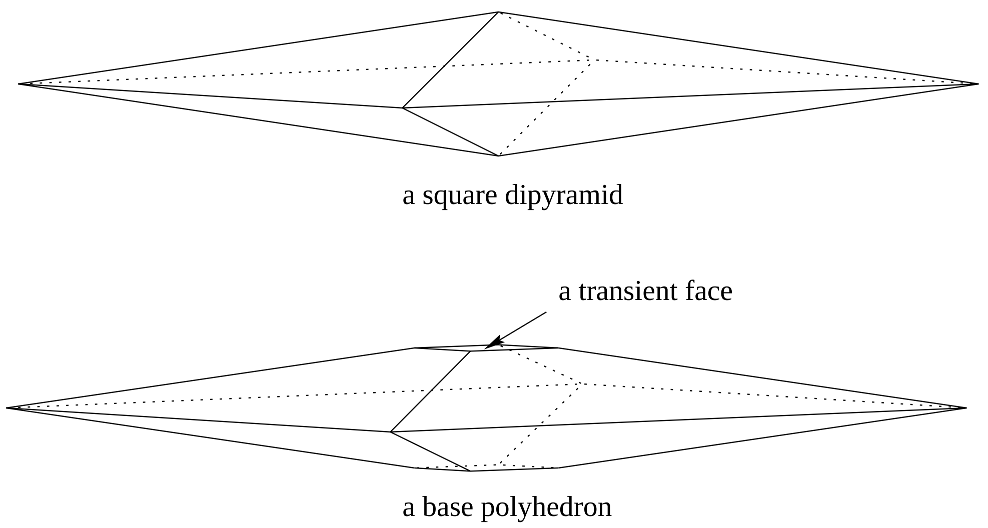


Figure 4.19: A square dipyramid and a base polyhedron created from that.

required is $O(N \log N)$.

Proof. We will give a proof sketch. First create a *base polyhedron* P, which is very "skinny" and is 5-visible face equiprojective, as follows. Take a very skinny square dipyramid (a square dipyramid is 4-visible face equiprojective) so that one pair of opposite apices is very close and the vertex angle at each of them is almost flat. Create two very small parallel quadrilateral faces f_1 and f_2 by chopping off these two apices. This is P. In Chapter 5 it is proven that the faces of a visible-face equiprojective polyhedron are in parallel pairs. Since a square dipyramid is 4-visible face equiprojective, adding parallel faces f_1 and f_2 makes it 5-visible face equiprojective. Observe that the dihedral angle at each edge of f_1 and f_2 is almost $180°$. Call f_1 and f_2 *transient.* (See Figure 4.19).

If all integers in S are five, then P is the resulting polyhedron. So assume that some integers are not five. Fix a position of P so that the directions of s are not parallel to its faces. Divide the directions of S into two groups S_1 and S_2 based on which transient face they see.

Consider the group S_1. If any integer in S_1 is not five, then replace f_1 by an unbounded polyhedron o_1 so that all directions in S_1 get the necessary number of additional visible faces from o_1. There may be several ways to create o_1. One way to do that is as follows. Place inside of P a small spherical ball b_1 in such a way that a tiny spherical cap is chopped off from b_1 by the plane of f_1. Remember that the dihedral angle at each edge of f_1 is almost $180°$. So b_1 is inside of P (i.e. b_1 does not intersect the faces adjacent to f_1). The radius of the sphere of b_1 is small enough so that if a similar (truncated) ball b_2 is placed on f_2, then b_1 and b_2 do not intersect. Note that f_1, which is the "entrance" of b_1, is made small enough so that for any two directions in S_1 their corresponding visible regions in b_1 do not overlap. Since there are finite number of directions in S_1, it is always possible to find such a small entrance of b_1 (or equivalently such a small f_1).

For each d-i pair $\langle d, n \rangle$ of S_1 compute the unique *visible region* r_d of d in the inner surface of b_1. The region r_d can be computed by projecting f_1 from d on b_1. Chop off a small spherical cap from b_1 such that the bounding circle c of that cap is strictly inside r_d. Now put vertices on c to create visible faces for d. If $n = 4$, simply put a vertex on c. If $n = 5$, create a triangle by taking three vertices on c. If $n > 5$, create $n - 4$ triangles as follows: take $n - 4$ vertices on c so that no two are opposite with respect to the center of c and if $n > 6$, then all of them are not within a half-circle of c; take another vertex v in the center of c and lift it slightly above the plane of c; finally connect v to the vertices on c. (See Figure 4.20).

After creating vertices and triangles for all d-i pairs in this way, take the convex hull of all of them including the vertices of f_1. The face f_1 will be in this convex hull and removing f_1 from it will give the resulting unbounded polyhedron o_1.

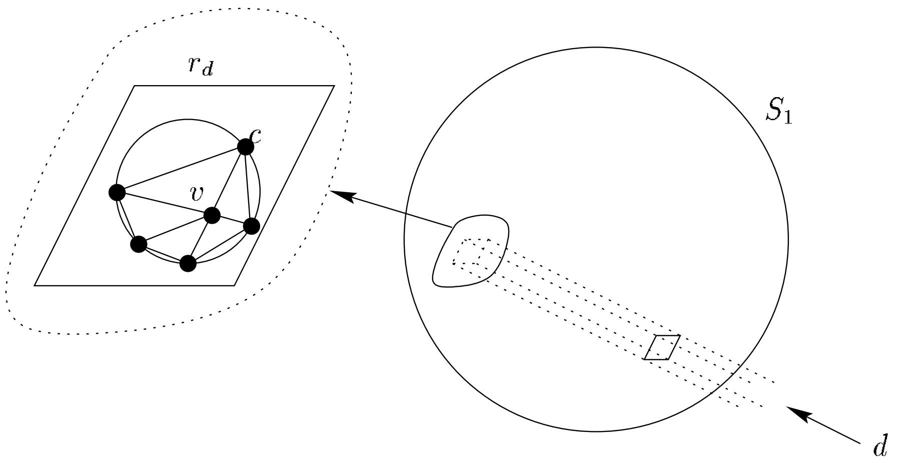


Figure 4.20: Creating visible faces of d in o_1.

Similarly create a an unbounded polyhedron o_2, if necessary, for the set S_2 by replacing f_2. This completes the construction of P.

Justification. Consider an arbitrary d-i pair $\langle d, n \rangle$ in S_1. When creating triangles in r_d, keep the height of the lifted vertex from the plane of c very small to ensure that all triangles and all vertices on c are in the convex hull. Then the faces of o_1 that are visible from d are exactly the triangles that have all three vertices on c or on its center. There are $n - 4$ such faces. Note that there are some faces in o_1 for which only one vertex is on c. These faces are partially visible, and thus, from the definition of a visible face, are invisible, from d. After f_1 is replaced by o_1, there are four other faces from the base polyhedron that are visible from d. So over all faces of P, the number of faces that are not parallel to d and are visible from d is n.

Finally, both o_1 and o_2 are inside of P and do not intersect each other. So P is simple and bounded too.

Now consider the time complexity and the size of P. Creating P as a base polyhedron and choosing its position takes $O(K)$ time. For each d-i pair $\langle d, n\rangle$ in S, finding the visible region, computing c, putting necessary vertices on, inside or above c, and finally creating triangles from them takes $O(n)$ time. Same can be done for all d-i pairs in $O(N)$ time. Finally, taking the convex hull twice take $O(N \log N)$ time, as the size of the convex hull is $O(N)$ in each case [19, 17]. With $K \in O(N)$, the total time is $O(N \log N)$.

The size of the polyhedron is linear to the total number of faces in o_1 and o_2, which are linear to the sum of all integers in S_1 and S_2. So the size of P is $O(N)$. $\square$

4.6 Perspective projections

This section briefly studies the problem of creating convex polyhedra from perspective projections. The differences between constructing polyhedra from perspective and orthogonal projections are similar to those for constructing polygons from perspective projections, which have already been discussed in Section 3.6. Such differences include that in perspective projections, the number of visible faces changes during the scaling and during the change of position of the feasible polyhedron, and that the volumetric size of a feasible polyhedron may be too small or too big.

We study a special case by assumng that the given view points are in the convex position. (The motivation behind this assumption has been discussed already in Section 4.6).

Each d-i pair in a d-i set consists of a view point d and an integer associated with d. A face e of a convex polyhedron is *visible* from a view point d if e faces towards d. More formally, e is visible from a view point p if and only if at some point x of e, the outward normal vector of e has a positive inner product with the vector from x to p.

4.6.1 Construction of the polyhedron

The main result for perspective projection is the following theorem.

Theorem 4.4 *Given a d-i set S of size $K \geq 3$, where all the view points are in convex position, then it is always possible to create a convex polyhedron P of size $N + O(1)$ such that for each d-i pair $\langle d, n \rangle$ in S the faces of P are not coplanar with d and the number of visible faces of P from d is n. Moreover, N is proportional to the sum of all integers in S.*

Proof. We give a proof sketch. For each view point d of S take a plane which is very close to d and separates d from all other view points. Take the half-space of this plane that does not contain d. Compute the intersection, which is convex, of all such half-spaces. If this intersection is unbounded, intersect additional half-spaces, which contain all view points of S, to make it bounded. Let the resulting convex polyhedron be P. For each view point d there is a face of P which comes from the corresponding plane for d and is visible only from d. Let this face be f. If $n > 1$ in the d-i pair $\langle d, n \rangle$, one can cut f as many times as necessary so that the total number faces created from f (including f) is n and all of them are visible only from d. (See Figure 4.21). Such a cut can be performed by a plane that separates d and the remaining part of P.

The size of P is N plus the number of extra half-spaces used to make P bounded, which should not be more than two. So the size of P is $N + O(1)$. $\square$

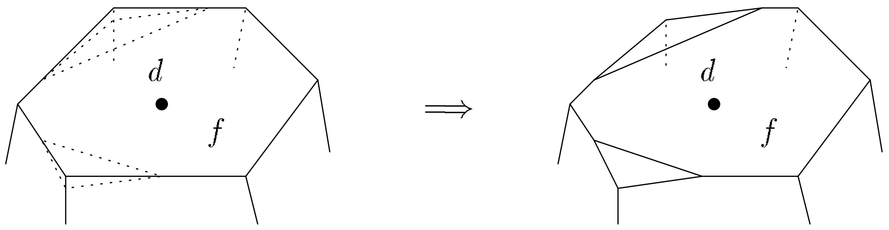


Figure 4.21: Increasing the number of visible faces for d (a) by cutting, and (b) by replacing the face f.

Chapter 5

Equiprojective Polyhedra

This chapter addresses the problem of reconstructing equiprojective polyhedra. As a first step of constructing equiprojective polyhedra, a characterization and an $O(n)$-time recognition algorithm for boundary-vertex equiprojective polyhedra are presented. Then it is proved that there are no 3- or 4-boundary vertex equiprojective polyhedra and that the triangular prism is the only 5-boundary vertex equiprojective polyhedron.

The notion of equiprojectivity from vertices in the projection boundary can be extended to visible vertices, edges, and faces in the projection. Under this extension, a characterization and $O(n)$ recognition algorithm of visible-face equiprojective polyhedra are shown. The most interesting result under this extension is that the generalized zonohedra are visible-face, visible-vertex, and visible-edge equiprojective. Finally, some relations among the different classes are discussed.

This chapter is organized as follows. After some preliminaries in Section 5.1, Section 5.2 contains the characterization and recognition of boundary-vertex equiprojective polyhedra. The minimality of boundary-vertex equiprojective polyhedra in

Section 5.3 and extend the definition of equiprojective polyhedra in Section 5.4.

5.1 Preliminaries

Throughout this chapter only convex polyhedra and their orthogonal projections are considered. Moreover, non-degenerate orthogonal projections are not considered, since in a degenerate projection a face f which is parallel to the view direction looks like an edge in the projection boundary and all the vertices and edges of f will coincide with that line. The reason for not considering perspective projections is that the concept of equiprojectivity does not hold for perspective projections, because the number of vertices, edges, and/or faces in the perspective projections as well as in their projection changes with the view point.

5.2 Boundary-vertex equiprojective polyhedra

Let us start with some examples— a cube is 6-boundary vertex equiprojective, a triangular prism[1] is 5-boundary vertex equiprojective, and a tetrahedron is not boundary-vertex equiprojective. Figure 5.1 shows two projections of the tetrahedron with different sizes of projection boundaries. Note that the cube and triangular prism can be generalized: for any $p \geq 3$, a prism based on a regular p-gon is $(p+2)$-boundary vertex equiprojective. An example of a boundary-vertex equiprojective polyhedron that is not a prism is given in Figure 5.3(a) on Page 110.

[1]A *p-gonal prism* consists of two parallel copies of a p-gon (the *bases*) with all other faces being parallelograms determined by the corresponding edges of the bases [46]. Note that oblique prisms are also included (i.e. not just right prisms).

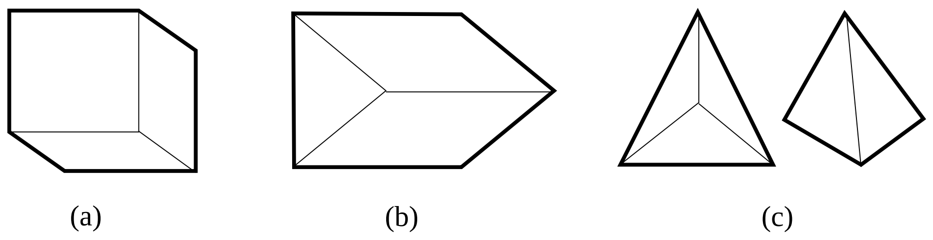


Figure 5.1: (a) A cube is 6-boundary vertex equiprojective, (b) a triangular prism is 5-boundary vertex equiprojective, and (c) a tetrahedron is not boundary-vertex equiprojective.

One way to test if a polyhedron is boundary-vertex equiprojective would be to check all the combinatorially different projections, which are called views, and for each of them count the number of edges of the projection boundary. Since the number views of an n vertex convex polyhedron is $O(n^2)$ and can be computed in $O(n^2)$ time (see Chapter 2), this method of testing for boundary-vertex equiprojectivity is inefficient. In this section we give a characterization of boundary-vertex equiprojective polyhedra, and show that this characterization provides an $O(n)$ time algorithm to test if a polyhedron of size n is boundary-vertex equiprojective.

Our characterization can be used to show that all generalized zonohedra are boundary-vertex equiprojective, and we identify other interesting subclasses as well (see Section 5.2.1). The whole class seems surprisingly rich, and we do not yet know of a method to generate all of its members.

5.2.1 Characterization

The flavor of the characterization is as follows. Any edge e of the projection boundary of P corresponds to some edge of P. As the view direction changes, e may leave the projection boundary. This only happens when a face f containing e in P becomes parallel to the direction. In order to preserve the size of the projection boundary, some other edge e' must join the projection boundary. In order for these events to occur simultaneously, e' must be an edge of f, or of a face parallel to f. This gives some intuition that the condition for equiprojectivity involves a pairing-up of parallel edge-face pairs of P. For a more precise statement of our characterization, see the following section.

Which polyhedra are boundary-vertex equiprojective?

We begin with the precise statement of the characterization, and then explore some classes of boundary-vertex equiprojective polyhedra.

For an edge e in face f, call (e, f) an *edge-face pair*. Two edge-face pairs (e, f) and (e', f') are *parallel* if e is parallel to e' and f is parallel to, or equal to, f'. Observe that in a convex polygon, an edge can have at most one parallel edge; and in a convex polyhedron, a face can have at most one parallel face. Thus an edge-face pair has at most three parallel pairs.

Define the *direction* of pair (e, f) to be a unit vector in the direction of edge e as encountered in a clockwise traversal of the outside of face f. The edge-face pairs (e, f) and (e', f') *compensate* each other if they are parallel and their directions are opposite (i.e. one is the negation of the other). In particular, this means that if $f = f'$, then e and e' are parallel (in which case they must be on “opposite sides” of f). On the other hand, if f and f' are distinct parallel faces, then e and e' are parallel edges lying on the “same side” of f and f', where by “same side” it means that the plane through e_1 and e_2 will have f_1 and f_2 in the same half-space. See Figure 5.2. An edge-face pair has at most two compensating pairs. Using this concept, we can characterize boundary-vertex equiprojective polyhedra.

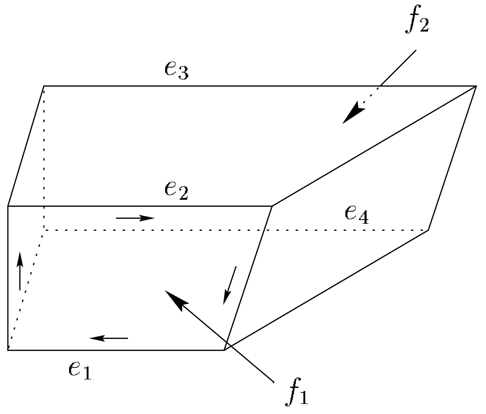


Figure 5.2: Example of some edge directions and compensating edge-face pairs. Direction of the edges of f_1 are shown. (e_1, f_1) is compensated by (e_2, f_1) and by (e_4, f_2) but not by (e_3, f_2).

Theorem 5.1 *Polyhedron P is boundary-vertex equiprojective if and only if its set of edge-face pairs can be partitioned into compensating pairs.*

Th proof of this theorem is given in the next section.

One of the simplest subclasses of boundary-vertex equiprojective polyhedra are the polyhedra where every face consists of parallel pairs of edges, i.e., a parallel-sided $2m$-gon. In this case an edge-face pair is compensated by the parallel edge in the same face. Note that generalized zonohedra fall in this class. (See Figure 5.3(b)).

For a zonohedron, since every face has a parallel pair with corresponding edges parallel, each edge-face pair could alternatively be compensated by the corresponding edge in the parallel face. More generally, we obtain the class of "face-compensating polyhedra", where any face not composed of parallel pairs of edges has a parallel face with corresponding edges parallel. The prisms based on odd regular polygons are in this class, but are not zonohedra.

Finally, there are boundary-vertex equiprojective polyhedra that are not face-

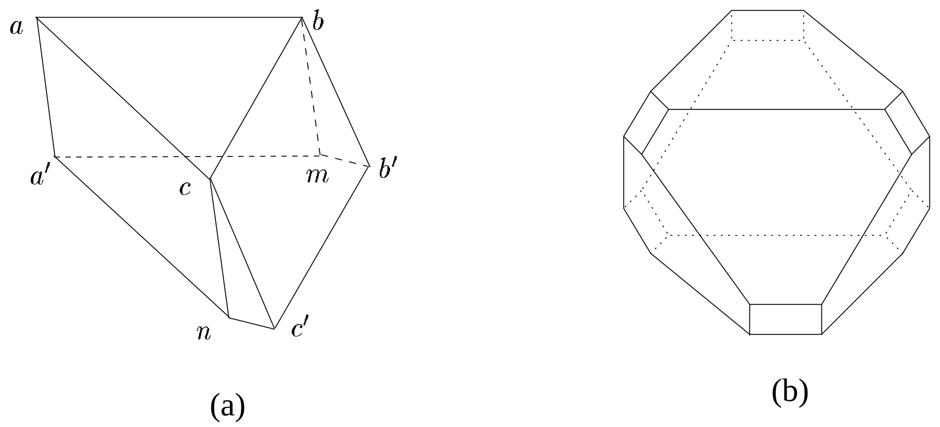


Figure 5.3: (a) A boundary-vertex equiprojective polyhedron which is not face compensating: the bottom face includes edges (m, b') and (n, c') that compensate each other, but the remaining edges are compensated by corresponding parallel edges in the top face (a, b, c). (b) A generalized zonohedron is boundary-vertex equiprojective.

compensating, for example the one shown in Figure 5.3(a).

5.2.2 Proof of characterization

Let P be a polyhedron. Given a view direction d, we can distinguish faces of P parallel to d, faces visible from d, and faces invisible from d. If there are no faces parallel to d, then the edges of the projection boundary of P projected in direction d are in one-to-one correspondence with the edges of P common to a visible and an invisible face of P. For a given direction d, let $\mathcal{S}_d$ be the set of edges of P that form edges of the projection boundary. As d changes continuously, $\mathcal{S}_d$ changes only when faces become parallel to d, on their way between visibility and invisibility or vice versa.

Note that if two faces of P are parallel to each other, then they both become parallel to d at the same time. The starting point is the claim that apart from such parallel faces, we can concentrate on the case where d crosses the plane of at most one face at a time.

Lemma 5.1 *We can change viewing directions continuously from any initial direction d_s to any other direction d_t, so that for any direction d along the way, the set of faces parallel to d is empty, or consists of one face—and its parallel counterpart if there is one. Furthermore, we can ensure that d crosses the plane of each face orthogonally in a small enough neighborhood.*

Proof. The set of all possible directions d corresponds to the set of points on a sphere. The directions parallel to a face f correspond to a great circle on the sphere, and the directions parallel to more than one face correspond to points on the sphere where two such circles intersect. There is a path on the sphere from any point d_s to any other point d_t that avoids the intersection points of two circles, and crosses circles orthogonally in a small neighborhood. □

Thus to show that a polyhedron is boundary-vertex equiprojective, it suffices to consider the changes in $\mathcal{S}_d$ as d orthogonally crosses the plane of one face f—and its parallel counterpart f' if it exists—causing f to become visible or invisible. Let $\mathcal{S}_d(f)$ be the edges of f that form edges of the projection boundary of P in direction d. We will use the notation $\mathcal{S}_d(f, f')$, where f and f' are always parallel, to mean $\mathcal{S}_d(f)$ together with $\mathcal{S}_d(f')$ if f' exists—i.e. the edges of f and f' that are edges of the projection boundary.

Take a direction d parallel to face f, but not parallel to any other face (except f' if it exists), let ε be a small vector normal to the plane of f, and consider the directions $d + \varepsilon$ and $d - \varepsilon$. These two directions make f invisible and visible, respectively, and affect no other faces except f' if it exists. For the polyhedron to be boundary-vertex equiprojective, $\mathcal{S}_{d+\varepsilon}(f, f')$ and $\mathcal{S}_{d-\varepsilon}(f, f')$ have the same cardinality.

Given a direction d parallel to face f, let the *upper chain* $U_d(f)$ of f with respect to direction d be the edges of f whose adjacent faces are visible from d, and let the *lower chain* $L_d(f)$ of f be all other edges of f. (See Figure 5.4(a)).

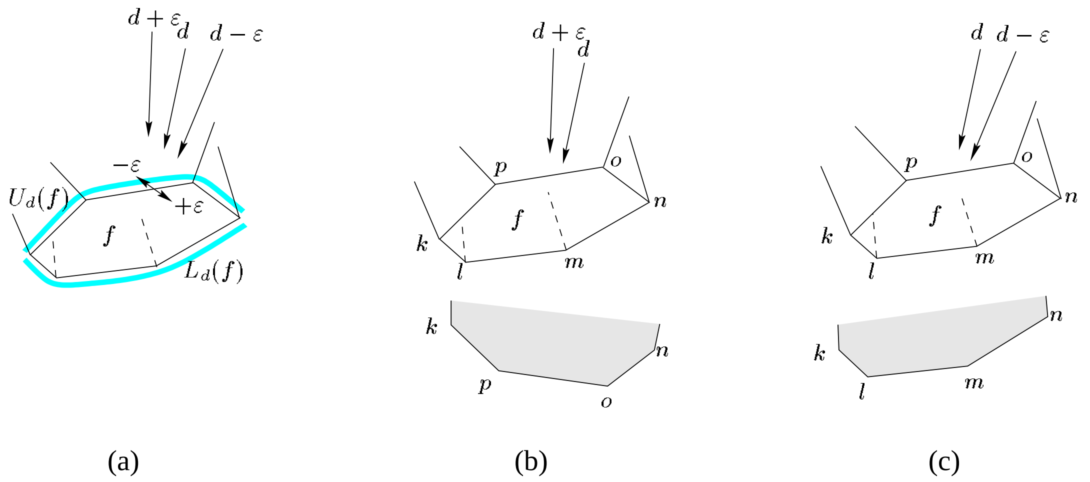


Figure 5.4: (a) Illustration of ε, and $U_d(f)$ and $L_d(f)$ for a face f; (b) $\mathcal{S}_{d+\varepsilon}(f) = U_d(f)$; (c) $\mathcal{S}_{d-\varepsilon}(f) = L_d(f)$.

Lemma 5.2 *Let d and ε be as above. Then $\mathcal{S}_{d+\varepsilon}(f) = U_d(f)$ and $\mathcal{S}_{d-\varepsilon}(f) = L_d(f)$.*

Proof. See Figure 5.4(b, c). $\square$

Corollary 5.1 *If f has a parallel face f' then $\mathcal{S}_{d+\varepsilon}(f') = L_d(f')$ and $\mathcal{S}_{d-\varepsilon}(f') = U_d(f')$.*

The above results give us the machinery we need to prove the sufficiency of the condition for equiprojectivity, and we now turn to a proof of the characterization.

Proof of Theorem 5.1

($\Longleftarrow$) Let P be a polyhedron whose edge-face pairs can be partitioned into compensating pairs. We will show that every orthogonal projection of P, except in directions parallel to faces, has the same number of edges.

By Lemma 5.1 it suffices to show that $\mathcal{S}_d$ maintains its cardinality as d orthogonally crosses the plane of one face f—and its parallel counterpart f', if it exists. Thus, using the notation above, we need to show that $\mathcal{S}_{d+\varepsilon}(f, f')$ and $\mathcal{S}_{d-\varepsilon}(f, f')$ have the same cardinality. By Lemma 5.2 and Corollary 5.1 we need to show that

$$|U_d(f)| + |L_d(f')| = |L_d(f)| + |U_d(f')|$$

Edge-face pairs involving f and f' can only be compensated by other edge-face pairs involving f and f'. Furthermore, an edge of $U_d(f)$ can only be compensated by an edge of $L_d(f)$ or $U_d(f')$, etc. Thus the fact that edge-face pairs can be partitioned into compensating pairs yields the equation above.

($\Longrightarrow$) Let P be a polyhedron whose edge-face pairs cannot be partitioned into compensating pairs. We will find two projections of P with different sizes.

Consider an edge-face pair (e, f). It lives in a "family" of at most 4 parallel edge-face pairs. An edge-face pair can only be compensated by others in its parallel family, and furthermore, can only be compensated by two others.

Consider the graph of compensating edge-face pairs, which has a vertex for each edge-face pair, and an edge when two pairs could compensate each other. The parallel family of (e, f) may consist of: (1) one node; (2) two isolated nodes; (3) two nodes joined by an edge; (4) three nodes joined in a path; (5) four nodes joined in a cycle. See Figure 5.5.

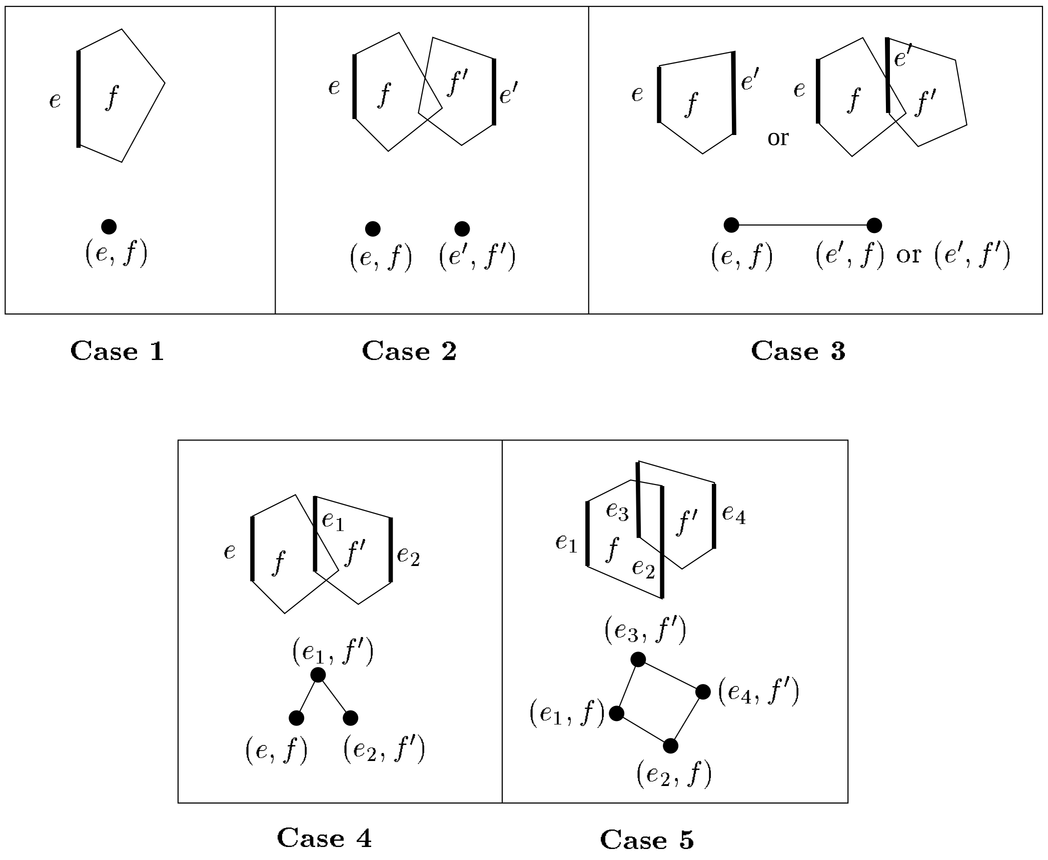


Figure 5.5: Graphs of compensating edge-face pairs within one parallel family. Faces f and f' are parallel. Edges drawn in bold are parallel.

In cases (3) and (5) the parallel family of (e, f) partitions into compensating pairs. In cases (1), (2), and (4) there is no partition into compensating pairs, and it must be shown that P is not boundary-vertex equiprojective.

Find a direction d in the plane of face f such that directions $d + \varepsilon$ and $d - \varepsilon$ yield projections of different sizes, for ε a small vector perpendicular to f. More precisely, choose d in the plane of face f (and its parallel counterpart f' if it exists) but not in the plane of any other face. For ε small enough, changes in the size of the projection between $d+\varepsilon$ and $d-\varepsilon$ are then due to the changes in f and f' only.

By Lemma 5.2 and Corollary 5.1 it suffices to choose d in the plane of f [and f'] but not in the plane of any other face so that:

$$|U_d(f)| + |L_d(f')| \neq |L_d(f)| + |U_d(f')|$$

We now have a 2-dimensional problem. Let c be the direction of edge e. Staying in the plane of f, let γ be a small vector perpendicular to e and directed to the outside of face f, and consider directions $c-\gamma$ and $c+\gamma$. We argue that, for one or the other, the above inequality holds. Then for γ small enough we can avoid the plane of any other face, and we have our value of d. See Figure 5.6(a).

Our argument is by contradiction. Suppose that we have equality for both $c-\gamma$ and $c+\gamma$. Then, rearranging to put f on the left hand side:

$$|U_{c-\gamma}(f)| - |L_{c-\gamma}(f)| = |U_{c-\gamma}(f')| - |L_{c-\gamma}(f')|$$

and

$$|U_{c+\gamma}(f)| - |L_{c+\gamma}(f)| = |U_{c+\gamma}(f')| - |L_{c+\gamma}(f')|$$

Subtracting yields

$$\begin{aligned}|U_{c-\gamma}(f)| - |U_{c+\gamma}(f)| + |L_{c+\gamma}(f)| - |L_{c-\gamma}(f)| = \\ |U_{c-\gamma}(f')| - |U_{c+\gamma}(f')| + |L_{c+\gamma}(f')| - |L_{c-\gamma}(f')|\end{aligned}$$

Now e is in $U_{c-\gamma}(f)$ but not $U_{c+\gamma}(f)$, and e is in $L_{c+\gamma}(f)$ but not $L_{c-\gamma}(f)$. There are no other changes due to f between $c-\gamma$ and $c+\gamma$. Thus the left hand side of the above equation is 2.

Now consider the right hand side. If f' doesn't exist or has no edges parallel to e (case (1)) then there are no changes between $c-\gamma$ and $c+\gamma$, i.e. $U_{c-\gamma}(f') = U_{c+\gamma}(f')$ and $L_{c+\gamma}(f') = L_{c-\gamma}(f')$, so the right hand side is 0. This is a contradiction.

If f' has two edges parallel to e (case (4)) then one is in U and one in L for each of $c-\gamma$ and $c+\gamma$, and, since there are no other changes, $|U_{c-\gamma}(f')| = |U_{c+\gamma}(f')|$ and $|L_{c+\gamma}(f')| = |L_{c-\gamma}(f')|$. Again, this makes the right hand side 0 and gives a contradiction.

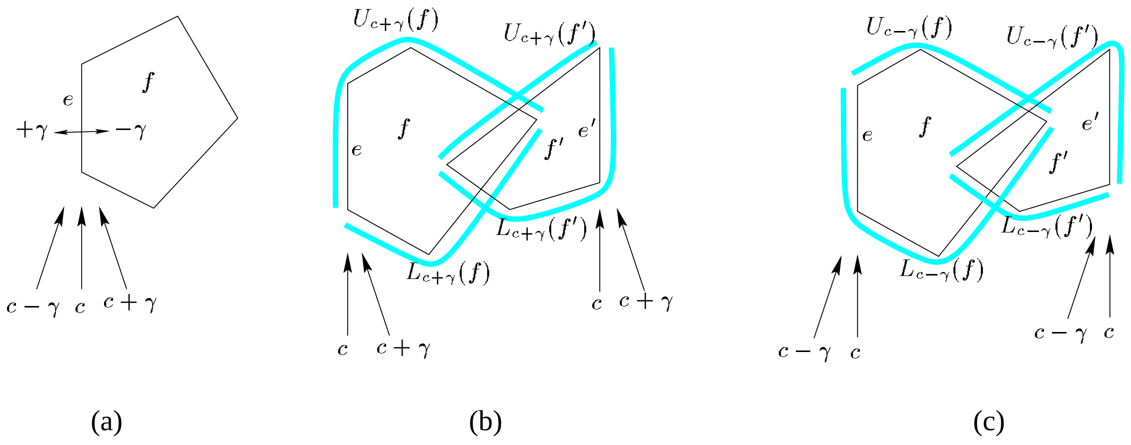


Figure 5.6: (a) Illustration of γ; (b,c) effect of γ on the number of edges in the projection boundary from two parallel faces in case (2).

Finally, if f' has one edge e' parallel to e (case (2)) then, since (e', f') does not compensate (e, f), they are on opposite sides of their faces, and e' is in $U_{c+\gamma}(f')$ but not $U_{c-\gamma}(f')$, and it is in $L_{c-\gamma}(f')$ but not $L_{c+\gamma}(f')$. See Figure 5.6(b,c). Since there are no other changes due to f' between $c-\gamma$ and $c+\gamma$, the right hand side of the above equation is -2. Contradiction. $\square$

5.2.3 Recognition algorithm

Our characterization provides an $O(n)$ time algorithm to test if a polyhedron is boundary-vertex equiprojective: There are $O(n)$ edge-face pairs. The pairs of parallel edges around a single face can be found by a clockwise or counter-clockwise scan using two pointers, one always diametrically opposite to the other. Similarly one can find the pairs of parallel faces by systematically exploring the diametrically opposite faces. Since each parallel family of edge-face pairs has at most 4 members, it is then a trivial matter to see if it can be partitioned into compensating pairs, see Figure 5.5.

Theorem 5.2 *An boundary-vertex equiprojective polyhedron can be recognized in $O(n)$ time.*

5.3 Minimal boundary-vertex equiprojective polyhedra

As a first step to explore the whole class of boundary-vertex equiprojective polyhedra, we will find the smallest boundary-vertex equiprojective polyhedra, where "smallest" is in terms of the number of vertices in the projection boundary. We prove that there are no 3 or 4-boundary vertex equiprojective polyhedra. Note that there is an arbitrary number of polyhedra which have some projection boundaries (but not all of them) with three or four vertices. We also prove that the triangular prism is the only 5-boundary vertex equiprojective polyhedron, and thus we call it the *smallest* boundary-vertex equiprojective polyhedron.

5.3.1 Proof of minimality

The idea of our minimality proof is as follows. We will first prove that any boundary-vertex equiprojective polyhedron P has at least one pair of parallel faces f and f'. Then we will argue that the two polygonal chains formed by the edges of f

and f' in the boundary of any projection of P are disjoint. Thus we will prove that the size of the projection boundaries of P is at least the total number of edges in the two smaller chains from f and f' plus two (for connecting the two chains to form the projection boundary polygon.) Then, because of f and f' have size at least three, it can be proved that f and f' together contribute at least three edges in the projection boundary. This will imply that the size of the projection boundary cannot be less than five. We now give the detailed proof.

Lemma 5.3 *Let P be a boundary-vertex equiprojective polyhedron. Then P has at least one pair of parallel faces.*

Proof. By way of contradiction assume that P does not have any parallel pair of faces. Then all the faces of P are self-compensating. In other words, all faces of P are parallel-sided $2m$-gons. But we know that a convex polyhedron all of whose faces are parallel-sided $2m$-gon is a generalized zonohedron and zonohedra have the property that for each face there is another parallel face, which is a contradiction. $\square$

By the above lemma, let f and f' be two parallel faces of P. Consider a view direction d in the plane of f and f' and not in the plane of any other face. Let the set of visible and invisible edges of f from d be $U_d(f)$ and $L_d(f)$ respectively. Let ε be a small outward normal vector to the plane of f. From Lemma 5.2 and Corollary 5.1, in the projection boundary of direction $d + \varepsilon$, the edges of f and f' are $U_d(f)$ and $L_d(f')$ respectively. Similarly in the projection boundary of direction $d - \varepsilon$, the edges of f and f' are $L_d(f)$ and $U_d(f')$ respectively.

Now we have the following lemma.

Lemma 5.4 *The two sets of edges in the projection boundary that correspond to $U_d(f)$ and $L_d(f')$, respectively, and similarly that correspond to $L_d(f)$ and $U_d(f')$, respectively, are vertex (and hence edge) disjoint.*

Proof. We prove this for $U_d(f)$ and $L_d(f')$ only. We know that the projection boundary of a convex polyhedron is a convex polygon. Since f and f' are parallel, they cannot have any common vertex or edge in P. This implies that in the projection boundary of P from direction $d + \varepsilon$, an edge of f is disjoint to an edge of f'. So $U_d(f)$ and $L_d(f')$ are disjoint. □

We are now ready for the main results regarding the minimal boundary-vertex equiprojective polyhedra.

Lemma 5.5 *Let P be k-boundary vertex equiprojective with two parallel faces f and f' respectively. Then neither f nor f' can have size more than $2k - 7$, where the size of a face is the number of edges in it.*

Proof. By Lemma 5.4, the projection boundary of P from each of the directions $d+\varepsilon$ and $d-\varepsilon$ contains two disjoint chains from f and f'. But any projection boundary of P is a convex polygon. So each of these two projection boundaries contains at least two more edges from the faces other than f and f'. As P is k-boundary vertex boundary-vertex equiprojective,

$$|U_d(f)| + |L_d(f')| \leq k - 2$$

and

$$|L_d(f)| + |U_d(f')| \leq k - 2.$$

After adding them together,

$$|U_d(f)| + |L_d(f')| + |L_d(f)| + |U_d(f')| \leq 2k - 4.$$

As the sum of $|U_d(f)|$ and $|L_d(f)|$ is the size of f, and similarly the sum of $|U_d(f')|$ and $|L_d(f')|$ is the size of f', the sum of the sizes of f and f' is no more than $2k-4$. So if f (similarly f') had size more than $2k-7$, then f' (similarly f) would have size less than three, which is impossible. $\square$

Corollary 5.2 *Let P be k-boundary vertex equiprojective. Then $k \geq 5$.*

Proof. Obviously k cannot be one or two. From Lemma 5.5 the size of two parallel faces f and f' of P are no more than $2k-7$. If k is three (four), then the sizes of f and f' are negative (less than 2), which is not possible for any face of P. $\square$

Theorem 5.3 *A triangular prism, which is defined as the two parallel copies of a triangle with corresponding vertices connected by edges, is the only 5-boundary-vertex equiprojective polyhedron.*

Proof. Let P be a 5-visible vertex equiprojective polyhedron. By Lemma 5.3, P has a pair of parallel faces, let them be f and f'. By Lemma 5.5 (with $k = 5$), f and f' are triangles. Let the counter-clockwise vertex-edge sequences of f and f' be $v_1, e_1, v_2, e_2, v_3, e_3$ and $v'_1, e'_1, v'_2, e'_2, v'_3, e'_3$ respectively. Since no edge in a triangular face can be compensated by an edge of the same face, each edge of f must be compensated by an edge of f'. So assume that e_1, e_2 and e_3 of f are compensated by e'_1, e'_2 and e'_3 of f' respectively. We have the following claim.

Claim 5.1 *For $1 \leq i \leq 3$, e_i and e'_i are two opposite edges of a parallelogram.*

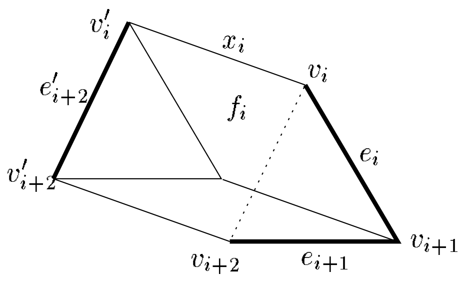


Figure 5.7: Illustrating the proof of Theorem 5.3.

Proof. We know that for $1 \leq i \leq 3$, e_i is parallel to e'_i and they are in the same side. So assume that for some $1 \leq i \leq 3$, $U_d(f) = \{e_i, e_{i+1}\}$, $L_d(f) = \{e_{i+2}\}$, $U_d(f') = \{e'_i, e'_{i+1}\}$, and $L_d(f') = \{e'_{i+2}\}$. This gives the counter-clockwise vertex-edge sequence of the chain formed by $U_d(f)$ as $v_i, e_i, v_{i+1}, e_{i+1}, v_{i+2}$ and that for $L_d(f')$ as v'_i, e'_{i+2}, v'_{i+2}. (See Figure 5.7).

From Lemma 5.4, $U_d(f)$ and $L_d(f')$ in the projection boundary of P from the direction $d+\varepsilon$ are disjoint. As the sum of $|U_d(f)|$ and $|L_d(f')|$ equals three, there are exactly two more edges. So v_i is connected to v'_i by an edge and v_{i+2} is connected to v'_{i+2} by another edge. In general, each edge e_i is connected with e'_i by two edges and thus form a polygon. Let this polygon be f_i. As e and e' are parallel, they and their two connecting edges are coplanar. So f_i is planar.

Finally, we prove that f_i is a parallelogram. It was proved at the beginning of the theorem that the only parallel faces of P are triangles. In other words, we say that if a face of P has size four or more, then it cannot have a parallel face. So f_i must be self-compensating and thus a parallelogram. $\square$

From the above claim, we see that P is composed of a pair of triangles with corresponding edges parallel and connected by a parallelogram. Moreover, the two triangles are equal, since the corresponding edges are of equal length. So P

cannot be anything but a triangular, not necessarily right triangular, prism. As from Corollary 5.2 there is no 3- or 4-visible vertex equiprojective polyhedra, we conclude that the triangular prism is the smallest boundary-vertex equiprojective polyhedron. □

5.4 Generalization of equiprojectivity

In this section we will see 2D counter-part of and give relations among visible-vertex, visible-edge, and visible-face equiprojective polyhedra. We will also show how generalized zonohedra fall in these different types. But first, let us see an example— a cube is 7-visible vertex, 9-visible edge, and 3-visible face equiprojective polyhedron (see Figure 5.8(a)).

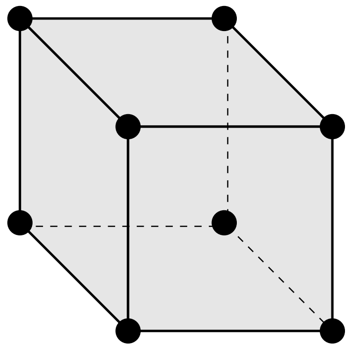


Figure 5.8: A cube is 7-visible vertex, 9-visible edge, and 3-visible face equiprojective polyhedron.

5.4.1 2D counter-part

What is the concept of equiprojectivity in 2D? The object we have to deal with is a convex polygon. The view directions are in the plane of the polygon. The projection of a polygon is always a line. The number of vertices in this projection boundary is always two, which are the two end vertices of the projected line. So under the actual definition of boundary-vertex equiprojective polyhedra, where the number of vertices in the projection boundary are the measure of equiprojectivity, all polygons are 2-boundary vertex equiprojective. Not that significant! If we consider the number of visible vertices and/or number of visible edges as the measure of equiprojectivity, then still it is easy to find such equiprojective polygons: They are parallel-sided $2m$-gons. Because in any projection of such a polygon, among each pair of parallel edges, one is visible and the other one is invisible (see Figure 5.9(a), where $m = 3$). On the other hand, a polygon P which is not a parallel-sided $2m$-gon has an edge e without any parallel pair, and P always has two projections of different size, where e is visible in one and invisible in the other while all other edges of P do not change their visibility, see Figure 5.9(b).

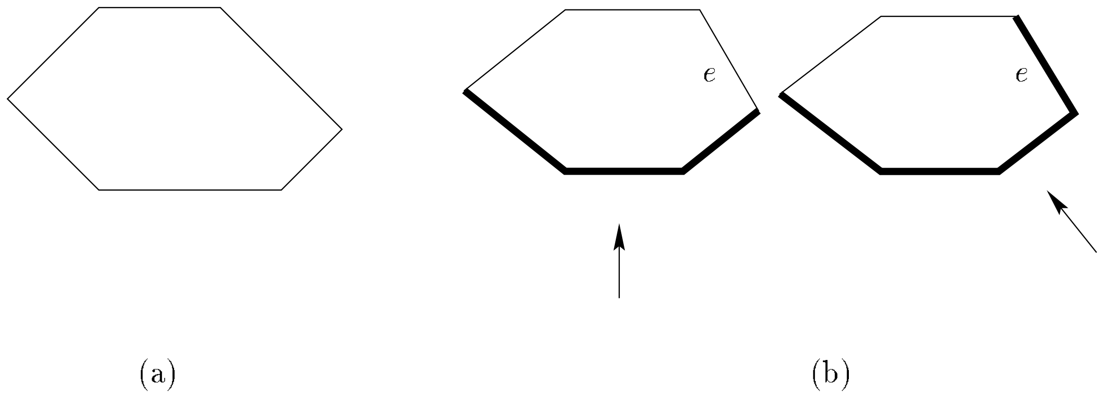


Figure 5.9: (a) A parallel-sided $2m$-gon, which is also visible-vertex and visible-edge equiprojective. (b) Two projections of a polygon, which is not a parallel-sided $2m$-gon. The lines in bold are the visible edges from corresponding directions shown by the arrows.

These new measures of equiprojectivity are more interesting in 3D and we study them in the remaining of this chapter. We start with finding relations among different types of equiprojective polyhedra.

5.4.2 Relation among different types

Observation 5.1 *(i) A boundary-vertex equiprojective polyhedron may not necessarily be visible-vertex, visible-edge, or visible-face equiprojective.*

(ii) If a polyhedron is equiprojective under any two of three measures— visible-vertex, visible-edge, and visible-face, the it is also equiprojective under the third measure.

Proof. (i) We show it by an an example. Let P be a prism. So P is boundary-vertex equiprojective. But P may have more than one projection with different number of visible vertices, visible edges, and visible faces. For example, Figure 5.10 shows two projections of a triangular prism where in one projection the number of visible vertices, visible edges, and visible faces are six, eight, and three, respectively, and in the other projection these numbers are five, six, and two, respectively.

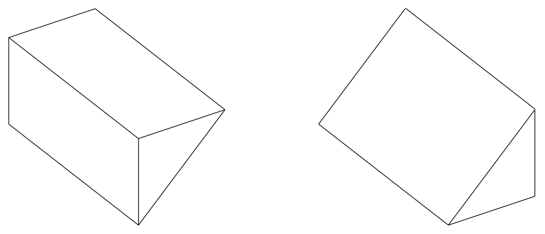


Figure 5.10: A prism which is not visible-vertex, visible-edge, or visible-face equiprojective.

(ii) Let P be a convex polyhedron. Observe that any projection of P is a plane graph[2]. From Euler's formula,[3] in any projection of P two among three numbers—the number of visible vertices, the number of visible edges, and the number of visible faces, are constant implies the third number is also constant. □

Our next observations are related to the classification of generalized zonohedra in different types of equiprojective polyhedra.

5.4.3 Generalized zonohedra

We first show that generalized zonohedra are visible-face equiprojective, although the reverse direction is not true— there are visible-face equiprojective polyhedra that are not zonohedra.

Observation 5.2 *(i) A convex polyhedron P is visible-face equiprojective if and only if each face of P has a parallel face; moreover P can be recognized in $O(n)$ time.*

(ii) A generalized zonohedron is visible-face equiprojective.

(iii) There are visible-face equiprojective polyhedra that are not zonohedra.

Proof. (i) While rotating P arbitrarily, whenever a face f of P moves from visible to invisible, another previously invisible face f' should become visible. This can happen if and only if f and f' are parallel. So all the faces of P should come in parallel pairs.

[2]A plane graph is a drawing of a planar graph where the edges are allowed to cross each other only at their end-vertices.

[3]For any plane graph with v vertices, e edges, and f faces, $v - e + f = 2$.

For recognizing P, one can find the pair of parallel edges in $O(n)$ time in the same way as described in Section 5.2.3.

(ii) Faces of a zonohedron come as parallel pairs.

(iii) For example consider the polyhedron P which is created from a cube by chopping off two vertices of each diagonal and thus creating four pairs of parallel triangles. See Figure 5.11. Now every face of P has a parallel pair, so P is visible-face equiprojective. But a zonohedron cannot have a triangular face. $\square$

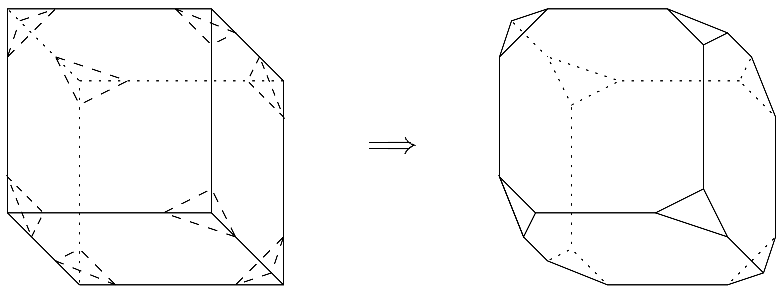


Figure 5.11: A pair of parallel triangles are created at the two ends of each diagonal of a cube.

Our most interesting observation regarding generalized zonohedra is that they are visible-edge and visible-vertex equiprojective. To our knowledge this property of generalized zonohedra, has not been indicated by anyone before.

Theorem 5.4 *Generalized zonohedra are visible-vertex and visible-edge equiprojective.*

Proof. Let P be a generalized zonohedron. First we prove that it is visible-edge equiprojective. P is boundary-vertex equiprojective. So for all projections of P the number of edges in the projection boundary is the same, let it be k. Let e be an edge defined by two faces f_1 and f_2. As the faces of P are in parallel pairs, each of f_1 and f_2 has a parallel pair. Let f_1' and f_2' be these two parallel faces respectively. Let e' be the edge defined by f_1' and f_2'. Now, in any projection of P, exactly one among f_1 and f_1' and one among f_2 and f_2' is visible. So if e is in the projection boundary, then e' is also in the projection boundary; moreover, if e is not in projection boundary but visible, then e' is also not in projection boundary but invisible. It means that for any projection of P, the number of visible edges that are not in the projection boundary is equal to the number of invisible edges that are also not in the projection boundary. Let this number be l. So P is $(k+l)$-visible edge equiprojective.

From Observation 5.2, P is visible-face equiprojective and then from Observation 5.1, P is visible-vertex equiprojective. $\square$

Chapter 6

View Point Selection

This chapter studies how to select view points of convex polyhedra such that the silhouette satisfies certain properties. Specifically, it gives algorithms to find all projections of a convex polyhedron such that a given set of edges, faces, and/or vertices appear on the silhouette. Remember that "projection boundary" and "silhouette" are the same for convex polyhedra, and the term "silhouette" will be used throughout this chapter.

We present algorithms to solve this problem in $O(k^2)$ time for k edges and in $O(n^2\alpha(n))$ time for k edges, vertices and faces, where $\alpha(n)$ is the inverse Ackerman function. For orthogonal projections, we give an improved algorithm that is fully adaptive in the number l of edge-disjoint paths formed by the given edges, and has a time complexity of $O(k \log k + kl)$. We then generalize this algorithm to vertices and/or faces appearing on the silhouette. Finally we study how to hide edges, faces, and/or vertices from the silhouette.

This chapter is organized as follows. After some preliminaries in Section 6.1, we deal with perspective projections in Section 6.2 and orthogonal projections in

Section 6.3. We study hiding in Section 6.4.

6.1 Preliminaries

Throughout this chapter we consider only convex polyhedra. We assume that the origin o is inside of the polyhedron. Also note that we do not consider an edge to be on the silhouette if its projection is the degenerate case of a single point. We will study how to find all view points for which a given set of edges, face, and/or vertices is on the silhouette. Note that the set of such view points may consists of more than one viewing regions.

We further investigate, in a different way, when exactly is an edge on the silhouette. Assume edge e is incident to faces f_1 and f_2 which are defined by half-spaces h_1 and h_2 with supporting planes π_1 and π_2 and containing the polyhedron. Remember that for $i = \{1, 2\}$, face f_i is visible from view point p if and only if p is not in half-space h_i (recall that the origin is inside the polyhedron and hence belongs to all half-spaces of all faces). Edge e is on the silhouette if and only if exactly one of its incident faces is visible from p, so p must be in exactly one of the half-spaces h_1 and h_2. Thus, p belongs to $(h_1 \cap \overline{h_2}) \cup (h_2 \cap \overline{h_1})$ (or more precisely, to the maximal open set contained within this set), which is a double-wedge formed by planes π_1 and π_2.

6.2 Perspective projections

In this section, we study how to find efficiently all view points of a convex polyhedron such that a given set of edges, vertices and/or faces is on the silhouette under perspective projections. Some of our results rely heavily on duality theory and transversal theory, which we review in Section 6.2.1.

We would like to recall the *reverse search* technique by Avis and Fukuda [5]. Reverse search is an exhaustive searching technique which can be implemented for various object-enumeration applications, including enumerating all cells in an arrangement of hyper-planes [5]. However, this technique is not directly applicable to our problem. The reverse search algorithm expects at least one object among all that are to be enumerated to be given. For our problem, finding an initial object which would be one of the resulting cells is not easy and its cost is comparable to computing the entire solution in the case when only edges are given and they form a single path. Moreover, in general the time complexity of reverse search is at least $O(d \cdot a \cdot o)$, where d is the maximum degree of "adjacency" among the objects, a is the time required for finding an adjacent object of a given one, and o is the output size (see [5] for details). In an implementation of reverse search for our problem for k edges, the value of d and o would be k and k^2, respectively, leading to a time complexity of at least $O(k^3)$.

On the other hand, a straightforward approach to find all view points from which a set $\mathcal{E}$ of edges is on the silhouette is to compute all possible views, or more efficiently, to compute only the views that are defined by the intersection of all the double-wedges associated with the edges in $\mathcal{E}$. In general, the number of views of convex polyhedron for perspective projections are $\theta(n^3)$ [66]; moreover, under perspective projections the arrangement of k double-wedges may result in $\theta(k^3)$ views. Therefore, finding all projections with certain properties can be done in $O(k^3 T)$ time, where T is the time to check whether a projection from a given viewing region has the desired property. In what follows we improve on this.

6.2.1 Geometric duality and transversal theory

Given a point $p = (a, b, c)$, the dual of the point is a plane $dual(p) = \{(x, y, z) : ax + by + cz = 1\}$. For a plane $\pi = \{(x, y, z) : ax + by + cz = d\}$, the dual is a point $dual(\pi) = (a/d, b/d, c/d)$. If π passes through the origin, then $dual(\pi)$ is at infinity. Recall that a point p lies in plane π if and only if $dual(\pi)$ lies in $dual(p)$. We will make use of a simple observation.

Lemma 6.1 *Let π be a plane that does not contain the origin o, and let p be a point that is not the origin o. Then π intersects the line segment $[o, p]$ if and only if $dual(p)$ intersects the line segment $[o, dual(\pi)]$.*

Proof. Let $\pi = \{(x, y, z) : ax + by + cz = 1\}$ and $p = (p_x, p_y, p_z)$. Plane π intersects $[o, p]$ if and only if $ap_x + bp_y + cp_z \geq 1$, which in turn holds if and only if the plane $\{(x, y, z) : p_x x + p_y y + p_z z = 1\}$ has o on one side and (a, b, c) on the other side, thus $dual(p)$ intersects $[o, dual(\pi)]$. $\square$

For an edge e, the dual is defined as the line segment $[dual(\pi_1), dual(\pi_2)]$, where π_1 and π_2 are the planes supporting the two incident faces of e. Pop et al. [67] made the following crucial observation.

Lemma 6.2 [67] *An edge e of a convex polyhedron is on the silhouette from view point p if and only if $dual(p)$ intersects $dual(e)$.*

A *geometric transversal* to a family of convex set in $\mathbb{R}^d$ is an affine subspace, such as a point, line, plane, or hyper-plane, that intersects every member of the family. Geometric transversal theory, more familiarly in two and three dimensions, concerns the complexity and efficient computation of the space of all [point, line, plane, or hyper-plane] transversals. See [31, 39] for detailed discussions on geometric transversal theory. We will use the following result:

Theorem 6.1 ([31], see also [39], Theorem 5.6) *Let $\mathcal{C}$ be a family of compact convex polytopes in 3D with a total of n vertices. All plane transversals of $\mathcal{C}$ can be found in $O(n^2\alpha(n))$ time, where $\alpha(n)$ is the inverse Ackerman function. If $\mathcal{C}$ consists of n line segments, then all plane traversals can be found in $O(n^2)$ time.*

We combine duality with transversal theory. Interestingly enough, Theorem 6.1 in turn uses dual geometric space (thus returning to the primal space) and analyzes double-wedges; it would thus be possible to express our algorithm directly in terms of double-wedges by tracing the results from transversal theory.

6.2.2 View point selection algorithm

Lemma 6.2 characterizes when an edge is on the silhouette of a projection. Combining this with transversal theory gives an algorithm to find all projections with a given set of k edges in $O(k^2)$ time. We apply the same approach to obtain an algorithm for given sets of vertices and faces. We first need to clarify what it means for a vertex or a face of a convex polyhedron to be on the silhouette. This is relatively straightforward for a vertex, which is on the silhouette if and only if two incident edges are on the silhouette. The notion of a face being on the silhouette is not entirely obvious, since the silhouette by nature consists of line segments, so the entire face cannot be on it. However, for the purpose of displaying the face "near" the silhouette, the following definition seems appropriate: A face f is considered to be on the silhouette from view point p if and only if f is visible from p and at least one edge of f is on the silhouette.

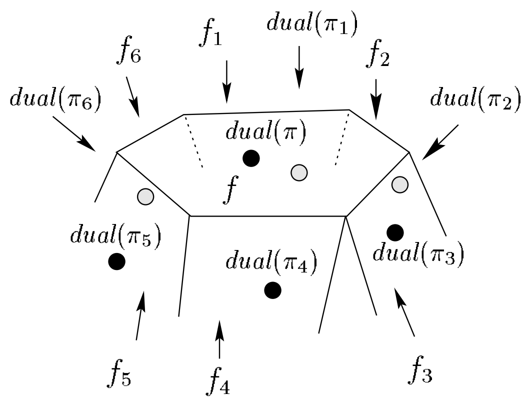
One might ask why "at least one edge " instead of "at least one vertex" is used in the above definition of a face on the silhouette. We decided on this since a vertex which is on the silhouette can have unknown number of adjacent visible faces and thus does not uniquely define a face on the silhouette, while an edge which is on the silhouette has exactly one adjacent visible face.

As in Lemma 6.2, we characterize when a vertex or a face is on the silhouette. Assume that v is a vertex, and let $f_1, \ldots, f_l$ be the faces, in circular order, adjacent to the vertex v. In dual space, the dual of v is a plane and the duals of the planes supporting $f_1, \ldots, f_l$ are points in $dual(v)$. So the dual points of the planes of $f_1, \ldots, f_l$ are co-planar and thus form a polygon, which we call the *dual polygon* associated with vertex v. Observe that this dual polygon is convex since we assume that the origin is inside the polyhedron.

Lemma 6.3 *A vertex v is on the silhouette from view point p if and only if its associated dual polygon is intersected by $dual(p)$.*

Proof. The polygon associated with v consists of the union of the dual of the edges incident to v. If $dual(p)$ intersects this polygon, then it intersects exactly two edges incident to v. These two edges are then on the silhouette, and in consequence, v is also on the silhouette. □

For a face f, proceed as follows: Let $f_1, f_2, ..., f_l$ be the faces adjacent to f and let $\pi, \pi_1, \ldots, \pi_l$ be the planes that support $f, f_1, \ldots, f_l$, respectively. Define the *dual polyhedron* of face f to be the convex hull of $dual(\pi)$, $dual(\pi_1), \ldots, dual(\pi_l)$ (see Figure 6.1).



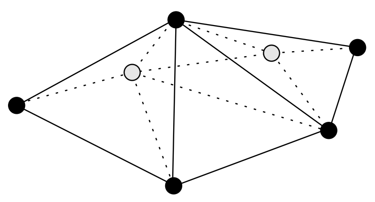


Figure 6.1: The dual polyhedron associated with a face f. (a) All necessary faces, and dual vertices of their supporting planes. (b) The resulting dual polyhedron.

Lemma 6.4 *A face f with supporting plane π of a convex polyhedron is on the silhouette from view point p if and only if $dual(p)$ intersects both the line segment $[o, dual(\pi)]$ and the dual polyhedron associated with f.*

Proof. Let P' be the dual polyhedron associated with f. This polyhedron has one vertex v_f for f, which is adjacent to all other vertices. Now, if $dual(p)$ intersects P', then it separates the vertices of P' into two groups and in particular intersects some of the edges incident to v_f, since all vertices are adjacent to v_f. Therefore, some of the edges of f are on the silhouette if and only if $dual(p)$ intersects P'.

For f itself to be on the silhouette, we additionally need that f is visible (i.e., not occluded) from view point p. This holds if and only if the plane through f separates the origin o (which is inside the polyhedron) from p. By Lemma 6.1, this holds if and only if $dual(p)$ intersects the line segment $[o, dual(\pi)]$. $\square$

Using the above lemmas, in combination with transversal theory, we can now compute all projections that have a given set of features on the silhouette.

Theorem 6.2 *Let P be a convex polyhedron with n vertices, and let $\mathcal{E}, \mathcal{V}$ and $\mathcal{F}$ be sets of edges, vertices and faces, respectively. All view points from which all edges in $\mathcal{E}$, all vertices in $\mathcal{V}$, and all faces in $\mathcal{F}$ are on the silhouette under perspective projections can be found in $O(n^2\alpha(n))$ time. The time reduces to $O(|\mathcal{E}|^2)$ if only edges are specified.*

Proof. Compute the dual edges of the edges in $\mathcal{E}$, the dual polygons associated with vertices in $\mathcal{V}$, and the dual polyhedra associated with faces in $\mathcal{F}$. Also, for each face $f \in \mathcal{F}$, compute the line segment $[o, \mathit{dual}(\pi)]$, where π is the plane supporting f. Now find all plane transversals that intersect these convex objects. By the above lemmas this gives exactly the perspective projections for which the features are on the silhouette.

Every edge creates one line segment to be transversed; by Theorem 6.1 we can find the projections for a set of edges in $O(|\mathcal{E}|^2)$ time. For a vertex v, the associated polygon has $\mathit{degree}(v)$ many vertices. So for a set of vertices, the total size of the associated polygons is no more than twice the number of edges in P, which is $O(n)$. Similarly for a face f, the associated polyhedron has $\mathit{degree}(f) + 1$ many vertices, and so for a set of faces, the total size of the associated polyhedra are no more than twice the number of edges plus the number of faces in P, which is $O(n)$. Hence, by Theorem 6.1 we can find all projections for a set of edges, vertices and faces in $O(n^2\alpha(n))$ time. $\square$

Transversal theory allowed us to force not only edges, but also vertices and faces, on the silhouette but at a higher cost since the size of the objects to be transversed is proportional to the degree of the vertex or face involved. Can the time complexities be improved in general? We leave this open for perspective projections.

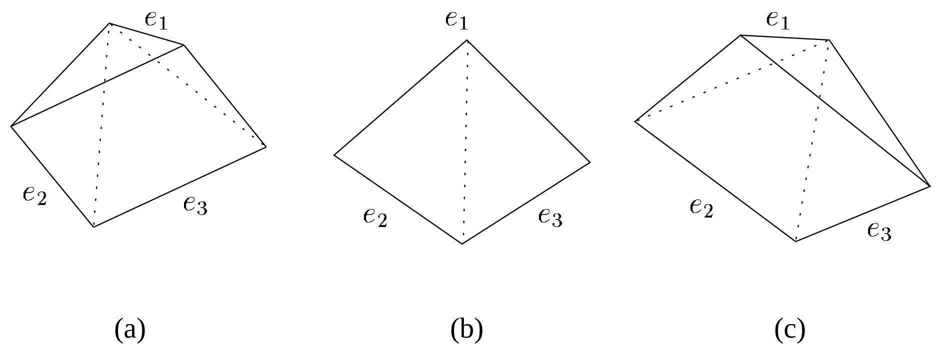


Figure 6.2: This polyhedron has more than two viewing regions with edges e_1, e_2 and e_3 on the silhouette. (a) One incident face of e_1 is visible. (b) A rotation such that e_1 reduces to a point; this view point is on the boundary between two viewing regions. (c) The other incident face of e_1 is visible.

We next consider the case when the set $\mathcal{E}$ contains only edges and all of them form a single path. For this case we give an improved algorithm of time $O(k \log k)$ and do not use duality theory.

6.2.3 Edges in one path

If a set of edges forms a path in the polyhedron, then it is easier to compute viewing regions for which they are on the silhouette, mostly because (as we will show) there are at most two viewing regions. This is not the case for arbitrary edges. For example Figure 6.2 shows a polyhedron where we have at least four viewing regions with edges e_1, e_2 and e_3 on the silhouette. Two views from two different viewing regions are shown in (a) and (c); the other two viewing regions are origin-symmetric to the ones illustrated here.

We now show that there are at most two regions from which the path can be on the silhouette.

Theorem 6.3 *Given a path of k edges on a convex polyhedron P, there are only two viewing regions from which all k edges of the path are on the silhouette of P, and we can find them in $O(k \log k)$ time.*[1]

Proof. Let the path consist of edges $e_1, \dots, e_k$. For $1 \leq i \leq k$, let f_i be the face to the left of edge e_i, where "to the left" is taken with respect to walking along the path from e_1 to e_k. Let f_i' be the other face incident to e_i. The crucial observation is that if the path is on the silhouette, then we either see all of $f_1, \dots, f_k$ or all of $f_1', \dots, f_k'$. To prove this, let v be the common vertex of e_1 and e_2. The clockwise order of faces around v is then f_1, (possibly) some other faces, f_2, f_2', (possibly) some other faces, f_1' (see Figure 6.3).

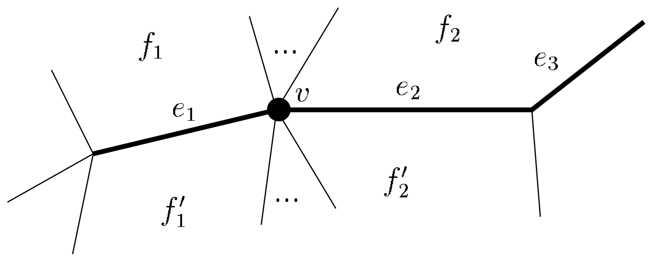


Figure 6.3: The ordering of faces around a vertex.

Since the visible incident faces of v are connected, but for each of $\{f_1, f_1'\}$ and $\{f_2, f_2'\}$ exactly one is invisible, there are only two possibilities: either all of $f_1, \dots, f_2$ are visible (and the others are invisible), or all of $f_2', \dots, f_1'$ are visible (and the others are invisible). So either $\{f_1, f_2\}$ are visible or $\{f_1', f_2'\}$ are visible, but it is not possible (for example) that f_1 and f_2' are visible. Assume f_1 and f_2 are visible. Repeating the argument for e_2 and e_3, this shows that f_3 must also be visible, and so on, so $f_1, \dots, f_k$ are all visible (and $f_1', \dots, f_k'$ are invisible).

[1]This theorem is implicitly assumed without proof in [14].

Alternatively, if f'_1 and f'_2 are visible, then all of $f'_1, \ldots, f'_k$ are visible (and $f_1, \ldots, f_k$ are invisible).

Recall that a face f is visible if and only if the view point is not in the half-space that defined the face. Thus, the view points from which $f_1, \ldots, f_k$ are all visible and $f'_1, \ldots, f'_k$ are all invisible are defined as the intersection of $2k$ half-spaces. This defines one viewing region. A second viewing region is the one from which $f'_1, \ldots, f'_k$ are all visible and $f_1, \ldots, f_k$ are invisible. This viewing region is again the intersection of half-spaces (in fact, the opposite half-spaces as those for the first viewing region).

Now, observe that any of these two viewing regions may be empty. It is easy to perceive when both regions are empty. An example when only one region is empty is shown in Figure 6.4. The polyhedron in this figure is a truncated pyramid. The set $\mathcal{E}$ consists of the edges of the smaller rectangular face f only. The only viewing region from which the edges of f are on the silhouette is the smaller pyramid that were truncated from the actual pyramid.

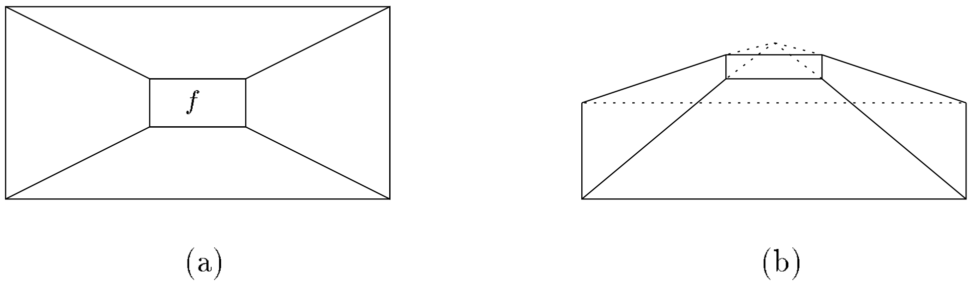


Figure 6.4: There is only one viewing region for all the edges of f to be on the silhouette, which is the complementary pyramid (shown in (b)) of the truncated pyramid in (a).

We now consider the time complexity. The half-spaces which form the viewing regions can be found in $O(k)$ time, and their intersection can be computed in $O(k \log k)$ time (see e.g. [68]). Thus, all viewing regions can be computed in $O(k \log k)$ time. $\square$

6.3 Orthogonal projections

In this section, we show how to compute efficiently all orthogonal projections such that a given set of edges is on the silhouette. This can be done with the same approach as in Theorem 6.1 (i.e., using duality theory and transversal theory.) However, a much simpler approach also works. Since the location of the origin is irrelevant in an orthogonal projection, we can identify directly the wedge for each edge and translate it such that all wedges intersect in one point. Then the hyper-plane arrangement defined by the k wedges has only $O(k^2)$ cells. Thus it can be possible to find all projections in $O(k^2 T)$ time, where T is the time to check whether a projection from a given viewing region has the desired property.

We improve on this further to give an algorithm that is adaptive in the number of paths formed by the set of edges. We study the case of many paths in Section 6.3.1. The ability to search all $O(k^2)$ cells of the arrangement allows us more flexibility in choosing projections; we study in Section 6.4 how to choose projections such that certain features (i.e. edges, vertices, faces) are *not* on the silhouette.

6.3.1 Edges in multiple paths

Note that Theorem 6.3 applies to orthogonal projections too, where the two viewing regions, after translated to the origin, are two opposite convex cones with their common at the origin and possibly both of them empty.

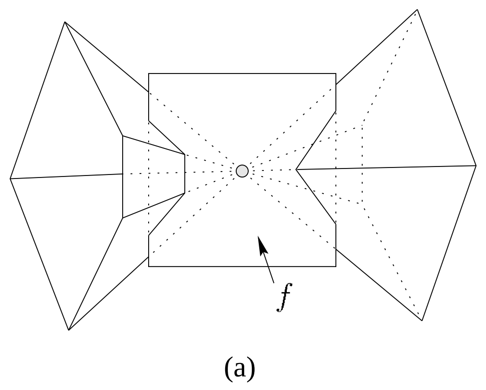
Corollary 6.1 *Given a path of k edges on a convex polyhedron P, for orthogonal projections there are exactly two viewing regions, possibly both of them are empty, from which all k edges of the path are on the silhouette of P, and we can find them in $O(k \log k)$ time.*

We now show how to use the above results to improve the time complexity in the case when k edges are not all disjoint. Assume that k edges are in l edge-disjoint paths $\mathcal{P}_1, \ldots, \mathcal{P}_l$; obviously $l \leq k$. From Corollary 6.1 we know how to compute all projections from which $\mathcal{P}_i$ is on the silhouette. We now show that for orthogonal projections, we can intersect these viewing regions in $O(k \log k + kl)$ time, which is an improvement over the $O(k^2)$ result of Section 6.2 if l is significantly smaller than k. As mentioned earlier, for orthogonal projections the set of view points from which e is on the silhouette is translation-invariant, since the view points are at infinity and correspond to directions. Hence, we are allowed to translate the double-wedge arbitrarily, and in particular we may assume that the intersection of the two planes contains the origin.

Theorem 6.4 *Given l disjoint paths of a convex polyhedron P, we can find all orthogonal projections of P such that all k edges of the paths are on the silhouette in $O(k \log k + kl)$ time.*

Proof. From Corollary 6.1 we know that for $1 \leq i \leq l$ there are exactly two viewing regions, say C_i^+ and C_i^-, with their common apex at the origin. Let $C_i = C_i^+ \cup C_i^-$ be the *double-cone* for path $\mathcal{P}_i$, and set $C^* = \bigcap C_i$; the desired viewing regions are then exactly the connected components of C^*.

In general, l double-cones may have $\theta(l^3)$ connected components in their intersection. For double-cones with origin at the apex this reduces to $O(l^2)$, but



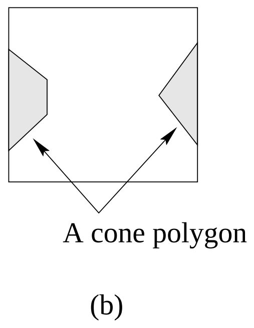


Figure 6.5: (a) Intersection of a double-cone C_i with a face f. (b) The corresponding cone polygon B_i has two disjoint components.

computing all connected components of C^* directly is still too slow. We therefore consider a projection of the double-cones onto a 2D surface (similar as in [9, 11]). Consider a unit cube D centered at the origin. To compute C^*, it suffices to compute the intersection of C^* with each face of D. We explain how to do this in the following for one face f of D only.

For any i, where $1 \leq i \leq l$, both C_i^+ and C_i^- can intersect f, but these intersections are disjoint (see Figure 6.5). Set $B_i = C_i \cap f$; then B_i is a single convex polygon or the union of two convex polygons. We call each B_i a *cone polygon*. Let $B^* = \bigcap B_i$, then each connected component of B^* corresponds to one viewing region.

To compute B^*, we compute the arrangement $\mathcal{A}$ of the cone polygons $B_1, \ldots, B_l$, which has at most $2l$ convex polygons with a total of at most $2k$ edges. This can be done in $O(k \log k + I)$ time where I is the number of intersection points [6, 18]. Within the same time bound, we can also compute the planar graph G defined by this arrangement (see Figure 6.6(a)). G has $O(k+I)$ vertices, edges and faces. Now we want to find all cells in the arrangement $\mathcal{A}$ that belong to all cone polygons. To do so, we compute a *modified directed dual graph* G' of G by computing the dual graph of G and replacing each edge by a directed 2-cycle (see Figure 6.6(b)). Note that here we use the term "dual" in the graph-theoretic sense, which is distinct from the geometric duality used before.

Each vertex in G' is a cell in $\mathcal{A}$ and each directed edge e in G' corresponds to entering or leaving a polygon of $B_1, \ldots, B_l$. We store with each edge of G' whether traversing this edge means entering or leaving a cone polygon. Finding all cells for which we are inside all l cone polygons can then be done by traversing the graph G' in such a way that all vertices are visited (e.g. with a DFS-traversal) and maintaining a counter of the number of cone polygons that the currently visited vertex is in. Since G' has $O(k+I)$ vertices and edges, this can be done in $O(k+I)$ time.

The time complexity of our algorithm hence is $O(k \log k + I)$. To find an upper bound on I, observe that there are $O(l)$ convex polygons of $O(k)$ edges total. Each edge can intersect each convex polygon at most twice, so $I \in O(kl)$ (and examples can be found where this is tight [4]). So the run-time of our algorithm is $O(k \log k + kl)$. $\square$

6.4 Computing projections for hiding features

Another application of view point selection is industrial design, where we might wish to make certain features easily visible or prominent, while some other features (such as service trap doors or unsightly wiring) should be hidden. Thus we would like to force edges, vertices or faces *not* to be on the silhouette; we say that these features are *hidden from the silhouette*. Note that for a convex polyhedron a vertex

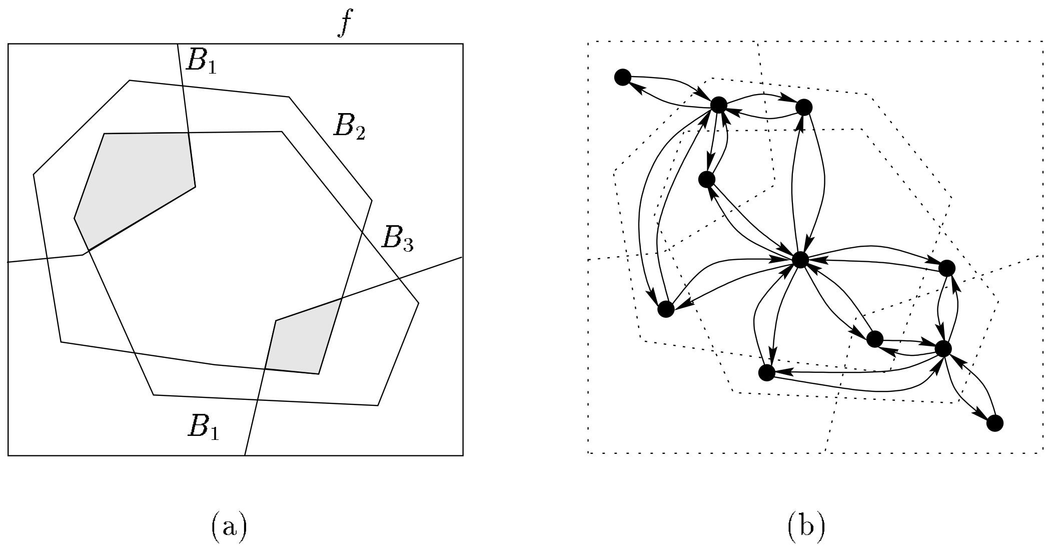


Figure 6.6: (a) The arrangement $\mathcal{A}$; the desired viewing regions are shown shaded. (b) The corresponding modified dual graph. Some edges and vertices have been omitted for clarity's sake.

is hidden from the silhouette if and only if all its incident edges are hidden from the silhouette. So for hiding a vertex from the silhouette it suffices to explain how to hide edges.

Lemma 6.5 *The set of all view points from which a set of k edges is not on the silhouette under orthogonal projections can be computed in $O(k^2)$ time.*

Proof. Each double-wedge from which an edge is on the silhouette also defines, by its complement, a double-wedge from which the edge is *not* on the silhouette. Since we are considering orthogonal projections, we can translate all double-wedges such that all hyper-planes that define them intersect the origin. Therefore, the hyper-plane arrangement now has only $O(k^2)$ cells and can be computed in $O(k^2)$ time. By using a traversal technique similar to the one in Section 6.3.1, we can check whether there is any cell in which all edges are hidden from the silhouette in $O(k^2)$ time. □

Corollary 6.2 *The set of all view points from which a set of vertices is not on the silhouette under orthogonal projections can be computed in $O(n^2)$ time.*

Proof. All k vertices are not on the silhouette if and only if all of their adjacent edges are hidden from the silhouette, and there are $O(n)$ such adjacent edges. □

Unlike a vertex, it is not true that a face is hidden from the silhouette if and only if all of its incident edges are hidden from the silhouette. Rather, a face is hidden from the silhouette if and only if it is invisible or it is visible and all of its incident edges are hidden from the silhouette. But this difference in the definitions is not very difficult to handle (see the following lemma).

Lemma 6.6 *The set of all view points from which a set of faces is not on the silhouette under orthogonal projections can be computed in $O(n^2)$ time.*

Proof. For each face f in the given set we have two cones from which f is not on the silhouette. One cone is defined by the whole half-space from which f is invisible (i.e. the half-space that does not define f.) The other cone corresponds to the view directions from which f is visible and all of its adjacent edges are not in the silhouette. This cone is computed as follows. For each adjacent edge e of f there is exactly one wedge from which f is visible and e is not in the silhouette. After translating all such wedges to the origin, their intersection gives the desired cone for f.

The time required to compute the second cone for f (as discussed above) is no more than $O(d \log d)$, where d is the degree of f, since each wedge is the intersection of two half-spaces [68]. So the total time for computing the pair of cones for all faces is $O(n \log n)$. After having all these pairs of cones we can compute their intersection, and then using a traversal technique similar to one in Section 6.3.1 we can find the cells in which all faces are not in the silhouette in $O(n^2)$ time. $\square$

Note that we can at the same time force some edges, vertices and faces on the silhouette and hide some other edges vertices and faces from the silhouette. With similar proofs as in the above lemmas and corollary, this can be done in $O(n^2)$ time.

Theorem 6.5 *Let P be a convex polyhedron with n vertices. Let $\mathcal{E}, \mathcal{E}'$ be sets of edges, $\mathcal{V}, \mathcal{V}'$ be sets of vertices, and $\mathcal{F}, \mathcal{F}'$ be sets of faces of P. All view points from which all elements of $\mathcal{E}, \mathcal{V}, \mathcal{F}$ are on the silhouette while all elements of $\mathcal{E}', \mathcal{V}', \mathcal{F}'$ are hidden under orthogonal projections can be found in $O(n^2)$ time. The time reduces to $O((|\mathcal{E}| + |\mathcal{E}'|)^2)$ if only edges are specified.*

Before we conclude this hiding section, we note that hiding edges unfortunately cannot easily be made adaptive in the number of paths that these edges form. The main obstacle is that Corollary 6.1 ("there are only two viewing regions") does not necessarily hold for hiding edges in one path. For example, consider the polyhedron in Figure 6.7 which is similar to the one in Figure 6.2. The path consisting of e_1 and e_2 is not on the silhouette from at least four viewing regions. Two views from two different viewing regions are shown in (a) and (c); the other two viewing regions are origin-symmetric to the ones illustrated here.

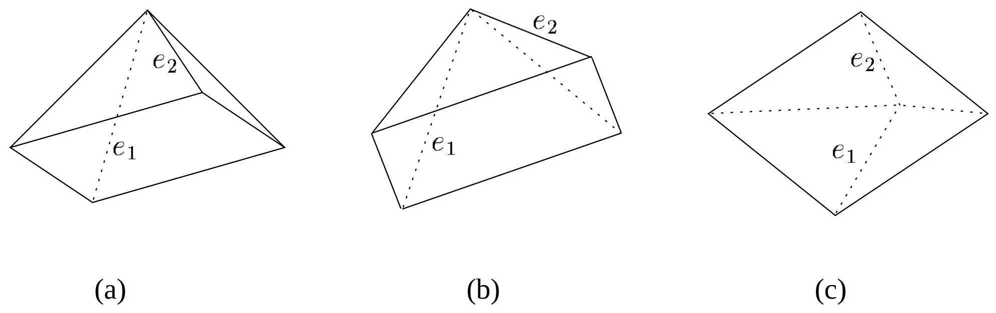


Figure 6.7: The polyhedron has more than two viewing regions with the path consisting of edges e_1 and e_2 not on the silhouette. (a) e_1 is invisible and e_2 is visible. (b) The path is partially not on the silhouette because of e_2 on the silhouette. (c) Both e_1 and e_2 are invisible.

Chapter 7

Conclusions and Open Problems

In this thesis we have studied the problems of reconstruction and visualization of polygons and polyhedra using their projections. In particular, we studied four types of problem: (i) reconstruction of polygons, (ii) reconstruction of polyhedra, (iii) equiprojective polyhedra, and (iv) visualization of polyhedra. Below is the summary of our results and open problems in each type.

Reconstruction of polygons. We have considered the problem of reconstructing a closed convex polygon given the number of visible edges from each of a set of orthogonal view directions. We have given necessary and sufficient conditions for the existence of a feasible polygon that realizes the projection and given an algorithm to construct one when it exists. We have also shown how to find the minimum and maximum size of a feasible polygon. We assumed that edges of a feasible polygon are not parallel to the view directions or collinear with the view points. Similar problems when we allow the edges to be collinear with view points can be considered as future work.

We have studied some special cases for non-convex polygons and for perspective projections. We have shown that we can always create a non-convex polygon when all directions see at least two edges. Here the size of the feasible polygon becomes proportional to the sum of the number of visible edges from all directions. We did not consider a partially visible edge as visible. Is it somehow possible to define the visibility of these edges and consider them in the creation of feasible polygons?

We briefly studied the problem of constructing convex polygons from perspective projections. We have shown that we can always construct a feasible polygon when the view points are in convex position. Here the size of the feasible polygon becomes equal to the sum of the number of visible edges from all directions. We think that it will be challenging to find feasible polygons when the view points are not in convex but arbitrary positions.

Finally, for both non-convex polygons and perspective projections it would be interesting to see algorithms for creating feasible polygons when the polygon size is given.

Reconstruction of polyhedra. We have studied creating polyhedra from projections. Here again our main focus was to create convex polyhedra from orthogonal projections. When the view directions are in one plane we proved that a feasible polyhedron can be created from a feasible polygon and vice versa, which implies that the necessary and sufficient condition for the existence of a feasible polyhedron is same as that for a feasible polygon. Like in 2D, we assumed that the faces of a feasible polyhedron are not co-planar with the view points. Problems similar to ones that we studied here can also be considered without this assumption.

Next we have considered the case when the view directions are covered by two planes. We have shown that the existence of a feasible polyhedron for each plane of directions separately is not enough for the existence of one feasible polyhedron for all directions. We have shown that when all directions see at least four faces, we can always create a feasible polyhedron if it exists. Our algorithm for creating a feasible polyhedron is by finding a valid assignment and a valid selection normal-points. For the case when the number of visible edges from the directions are not necessarily four or higher, we have shown that for all possible valid assignments there may not be any valid selection; it remains open to decide the existence of a feasible polyhedron for this case.

It would be interesting to see an algorithm to decide the existence of a feasible polyhedron when the view directions are covered by more than two planes. Our conjecture is that this problem is NP-hard.

We always assumed that the planes covering the directions of $\mathcal{S}$ are given. It is immediate to ask how fast can we do that?

Equiprojective polyhedra. We have studied equiprojective polyhedra as a special case of the problem of reconstruction of polyhedra where all projections have the same characteristics. We have given a characterization of boundary-vertex equiprojective polyhedra. From there we have given an $O(n \log n)$-time recognition algorithm to recognize boundary-vertex equiprojective polyhedra.

We have shown that there is no 3- or 4-boundary vertex equiprojective polyhedron and the triangular prism is the only 5-boundary vertex equiprojective polyhedron.

Finally we extended the definition of equiprojective polyhedra to visible-vertex, visible-edge, and visible-face equiprojective polyhedra. Among these three classes of polyhedra, we have given a characterization of visible-face equiprojective polyhedra and shown that such a polyhedron can be recognized in $O(n \log n)$ time. We have also shown some relations among these three classes and discovered that generalized zonohedra are visible-vertex and visible-edge equiprojective.

Several interesting questions remain to be answered for equiprojective polyhedra. We leave open the question of an algorithm to generate all boundary-vertex equiprojective polyhedra, or even to generate just the face-compensating ones. Note that there are algorithms to generate zonohedra [34]. Our most interesting example, the non-face compensating boundary-vertex equiprojective polyhedron in Figure 5.3(a), is formed by adjoining two prisms. Can all boundary-vertex equiprojective polyhedra be constructed in some way from zonohedra and other face-compensating polyhedra?

For visible-vertex, visible-edge, and visible-face equiprojective polyhedra we only discovered that generalized zonohedra fall into all of them. Is there any other polyhedron that fall in these three classes? We conjecture that the answers is no.

Finally what are the smallest visible-vertex and visible-edge equiprojective polyhedra?

Visualization of polyhedra. Finally we have studied the question of how to find all projections of a polyhedron that satisfy certain properties. We focused on how to force a given set of edges on the silhouette of a convex polyhedron, and gave an efficient algorithm to do so, as well as an adaptive algorithm for orthogonal projections if the edges are in one or very few connected paths. The most immediate open problem is whether such an adaptive algorithm exists for perspective projection. Also, can the time complexities be improved further?

We also briefly studied how to hide edges from the silhouette of a convex polyhedron. But here the time complexities are higher than that for forcing edges in the silhouette. The reasons are that the technique of transversal theory (Theorem 6.1) cannot be applied, since the resulting shapes are not convex and compact. Moreover, an adaptive algorithm cannot be considered here since Corollary 6.1 does not hold in this case. So how can we efficiently find all view points from which a given set of edges is hidden?

Finally, all our results were for convex polyhedra. What are efficient algorithms for computing projections of non-convex polyhedra such that the silhouette satisfies given properties?

Glossary

Chapters 3 and 4

$\bar{}$, $\tilde{}$: The symbols $\bar{}$, and $\tilde{}$ are used ober the symbols to distinguish them in their respective planes.

δ_i: The i-th view difference in a proper d-i set $\mathcal{S}$: $\mathbf{max}\{0, n_{i+1} - n_i\}$.

$\delta_{i,j}$: The view difference associated with the $\{i, j\}$-th d-polygon.

θ_i: The chord (lune) of circle c (sphere s), which is centered at the origin, that is invisible from d_i but visible from d_{i+1}.

$\theta_{i,j}$: The spherical polygon (also called a d-polygon) resulting from the intersection of the i-th and j-th d-lunes from two groups of $\mathcal{S}$.

Δ_i: The i-th final view difference in a proper d-i set $\mathcal{S}$.

$\Delta_{i,j}$: The final view difference associated with the $\{i, j\}$-th d-polygon.

C, R: A column and a row of a matrix.

$\langle d_i, n_i \rangle$: The i-th d-i pair of a d-i set according to a circular sequence of the directions.

D: Sum of all view differences in a proper d-i set $\mathcal{S}$.

$D(N)$: Sum of all view differences in a proper d-i set $\mathcal{S}(N)$ as a function of N when N is unknown.

h_i: The visible half-circle/hemisphere of d_i.

K: Number of directions (d-i pairs) in a d-i set S.

N: Size of the feasible polygon/polyhedron.

S: A d-i set.

$\mathcal{S}$: A proper d-i set.

$\mathcal{S}(N)$: A proper d-i set whose integers are function of N when N is unknown.

d-i pair $\langle d, n\rangle$: A d-i pair of a view direction or view point d and an integer n.

d-i set: A set of d-i pairs.

d-arc: The arc of the circle c, which is centered at the origin, that is invisible and visible from two consecutive directions of $\mathcal{S}$.

d-lune: The spherical lune of the sphere s, which is centered at the origin, that is invisible and visible from two consecutive directions of $\mathcal{S}$.

d-polygon: The spherical polygon resulting from the intersection of two or more d-lunes.

i**-th view difference:** Minimum number of edges (faces) of a feasible polygon (polyhedron) that are invisible from d_i but visible from d_{i+1}, also can be written as $\mathbf{max}\{0, n_{i+1} - n_i\}$.

i-th final view difference: The number of edges (faces) of a feasible polygon (polyhedron) that are invisible from d_i but visible from d_{i+1}.

$\{i,j\}$-th view difference: Minimum number of normal-points of a feasible polyhedron that should be within the $\{i,j\}$-th d-polygon $\theta_{i,j}$.

$\{i,j\}$-th final view difference: Exact number of normal-points of a feasible polyhedron that are within the $\{i,j\}$-th d-polygon $\theta_{i,j}$.

feasible polygon (polyhedron) for S A convex polygon (polyhedron) such that from each d-i pair $\langle d, n\rangle$ of S the the number of visible edges (faces) from d is n.

poly-sum: The sum of the final view differences of the d-polygons that are within a d-lune (similarly within a visible hemisphere).

proper d-i set: A set of d-i pairs where for each d-i pair $\langle d, n\rangle$ there is another d-i pair $\langle d', N-n\rangle$ such that the direction d' is opposite to the direction d.

Chapter 5

$\mathcal{S}_d$: The shadow of a convex polyhedron from direction d.

$L_d(f)$: The lower chain of f with respect to d: the set of edges of f whose adjacent faces other than f are invisible from d.

$U_d(f)$: The upper chain of f with respect to d: the set of edges of f whose adjacent faces other than f are visible from d.

compensating pair of edge-face pairs: A pair of edge-face pairs (e,f) and (e',f') such that e and e' are parallel and f and f' are either parallel or the same.

edge-face pair (e,f): An edge e and one of its adjacent faces f.

Bibliography

[1] P. K. Agarwal and M. Sharir. Applications of a new space partitioning technique. *Discrete and Computational Geometry*, 9:11–38, 1993.

[2] P. K. Agarwal and M. Sharir. On the number of views of polyhedral terrains. *Discrete and Computational Geometry*, 12:177–182, 1994.

[3] B. Aronov, H. Brönnimann, D. Halperin, and R. Schiffenbauer. On the number of views of polyhedral scenes. In *3rd Japanese Conference on Discrete and Computational Geometry*, pages 81–90, Tokyo, Japan, November 2000.

[4] B. Aronov and M. Sharir. The common exterior of convex polygons in the plane. *Computational Geometry: Theory and Applications*, 8(3):139–149, 1997.

[5] D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete and Applied Mathematics*, 6(1–3):21–46, 1996.

[6] I. J. Balaben. An optimal algorithm for finding segments intersections. In *11th ACM Symposium on Computational Geometry*, pages 211–219, Vancouver, British Columbia, Canada, June 1995.

[7] W. W. R. Ball and H. S. M. Coxeter. *Mathematical Recreations and Essays.* University of Toronto Press, Toronto, 1974.

[8] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali. *Linear Programming and Network Flows.* John Wiley, New Jersey, 2005.

[9] F. Benichou and G. Elber. Output sensitive extraction of silhouettes from polygonal geometry. In *7th Pacific Conference on Computer Graphics and Applications*, pages 60–69, Seoul, Korea, October 1999.

[10] S. Bereg. 3D realization of two triangulations of a convex polygon. In *20th European Workshop Computational Geometry*, pages 49–52, Seville, Spain, March 2004.

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

[12] A. Bottino, L. Jaulin, and A. Laurentini. Reconstructing 3D objects from silhouettes with unknown viewpoints: The case of planar orthographic views. In *8th Iberoamerican Congress on Pattern Recognition*, pages 153–162, Havana, Cuba, November 2003.

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

[14] P. Brunet, I. Navazo, J. Rossignac, and C. Saona-Vazquez. Hoops: 3D curves as conservative occluders for cell-visibility. *Computer Graphics Forum*, 20(3):431–442, 2001.

[15] T. Burger and P. Gritzmann. Finding optimal shadows of polytopes. *Discrete and Computational Geometry*, 24(2-3):219–240, 2000.

[16] T. Burger, P. Gritzmann, and V. Klee. Polytope projection and projection polytopes. *American Mathematical Monthly*, 103(9):742–755, 1996.

[17] T. M. Chan. Output-sensitive results on convex hulls, extreme points, and related problems. *Discrete and Computational Geometry*, 16(4):369–387, 1996.

[18] B. Chazelle and H. Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *Journal of ACM*, 39(1):1–54, 1992.

[19] B. Chazelle and J. Matousek. Derandomizing an output-sensitive convex hull algorithm in three dimensions. *Computational Geometry: Theory and Application*, 5(1):27–32, 1995.

[20] B. Chazelle, M. Sharir, and E. Welzl. Quasi-optimal upper bounds for simplex range searching and new zone theorems. *Discrete and Computational Geometry*, 8:407–429, 1992.

[21] V. Chvátal. *Linear Programming.* W. H. Freeman, New York, 1983.

[22] M. B. Clowes. On seeing things. *Artificial Intelligence*, 2(1):79–116, 1971.

[23] H. S. M. Coxeter. *Regular Polytopes.* Macmillan, New York, 1963.

[24] C. Croft, K. Falconer, and R. Guy. *Unsolved Problems in Geometry.* Springer-Verlag, New York, 1991.

[25] P. R. Cromwell. *Polyhedra.* Cambridge University Press, Cambridge, 1997.

[26] G. Das and M. T. Goodrich. On the complexity of optimization problems for 3-dimensional convex polyhedra and decision trees. *Computational Geometry: Theory and Applications*, 8(3):123–137, 1997.

[27] M. de Berg, D. Halperin, M. Overmars, , and M. van Kreveld. Sparse arrangements and the number of views of polyhedral scenes. *International Journal of Computational Geometry Applications*, 7(3):175–195, 1997.

[28] B. V. Dekster. Convex hulls of spatial polygons with a fixed convex projection. *Contributions to Algebra and Geometry*, 36(1):123–134, 1995.

[29] E. D. Demaine and J. Erickson. Open problems on polytope reconstruction. Manuscript, 1999.

[30] P. Eades, M. E. Houle, and R. Webber. Finding the best viewpoints for three-dimensional graph drawings. In *5th International Symposium on Graph Drawing*, pages 87–98, Rome, Italy, September 1998.

[31] H. Edelsbrunner, L. Guibas, and M. Sharir. The upper envelope of piecewise linear functions: Algorithms and applications. *Discrete Computational Geometry*, 4:311–336, 1989.

[32] A. Efrat, L. Guibas, O. Hall-Holt, and L. Zhang. On incremental rendering of silhouette maps of a polyhedral scene. In *11th ACM-SIAM Symposium on Discrete Algorithms*, pages 910–917, San Francisco, California, January 2000.

[33] D. Eppstein. The geometry junkyard: Zonohedra. `http://www.ics.uci.edu/~eppstein/junkyard/zono.html`.

[34] D. Eppstein. Zonohedra and zonotopes. *Mathematica in Education and Research*, 5(4):15–21, 1996. `http://www.ics.uci.edu/~eppstein/pubs/p-zono.html`.

[35] R. Gardner. *Geometric Tomography.* Cambridge University Press, Cambridge, 1995.

[36] Z. Gigus, J. canny, and R. Seidel. Efficiently computing and representing aspect graphs of polyhedral objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):542–551, 1991.

[37] Z. Gigus and J. Malik. Computing the aspect graph for line drawings of polyhedral objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2):113–122, 1990.

[38] F. Gomez, F. Hurtado, T. Sellares, and G. Toussaint. Nice perspetive projections. *Journal of Visual Communication and Image Representation*, 12(4):387–400, 2001.

[39] J. E. Goodman, R. Pollack, and R. Wenger. Geometric transversals theory. In J. Pach, editor, *New Trends in Discrete and Computational Geometry.* Springer-Verlag, 1993.

[40] G. W. Hart. Encyclopedia of polyhedra. `http://www.georgehart.com/virtual-polyhedra/vp.html`.

[41] M. Hasan and A. Lubiw. Equiprojective polyhedra. In *15th Canadian Conference on Computational Geometry*, pages 47–50, Halifax, canada, August 2003.

[42] F. S. Hillier and G. J. Lieberman. *Introduction to Operations Research.* McGraw-Hill Higher Education, Boston, 2005.

[43] C. H. Hoffman. *Geometric and Solid Modelling: An Introduction.* Morgan Kaufmann, California, 1989.

[44] D. A. Huffman. Impossible objects as nonsense sentences. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 6, pages 295–323, 1971.

[45] C. S. Kaplan and G. W. Hart. Symmetrohedra: Polyhedra from symmetric placement of regular polygons. In *Bridges 2001: Mathematical Connections in Art, Music and Science*, Winfield, Kensas, July 2001. `http://www.cgl.uwaterloo.ca/~csk/papers/bridges2001.html`.

[46] W. Karush. *The Crescent Dictionary of Mathematics.* The MacMillan Company, New York, 1962.

[47] H. S. Kasan and K. D. Kumar. *Introductory Operations Research: Theory and Applications.* Springer, Berlin, 2004.

[48] L. Kettner and E. Welzl. Contour edge analysis for polyhedron projections. In W. Strasser, R. Klein, and R. Rau, editors, *Geometric Modeling: Theory and Practice*, pages 379–394. Springer, 1997.

[49] J. J. Koenderink. What does the occluding contour tell us about solid shape? *Perception*, 13(3):321–330, 1984.

[50] M. R. Korn and C. R. Dyer. 3D multiview object representations for model-based object recognition. *Pattern Recognition*, 20(1):91–103, 1987.

[51] A. Laurentini. How many 2D silhouettes does it take to reconstruct a 3D object? *Computer Vision and Image Understanding*, 67(1), 1997.

[52] H. Lipson and M. Shpitalni. Optimization-based reconstruction of a 3D object from a single freehand line drawing. *Computer Aided Design*, 28(8):651–663, 1996.

[53] L. A. Lyusternik. *Convex Figures and Polyhedra.* Dover, New York, 1963. Translated from Russian by T. J. Smith.

[54] A. K. Macworth. Interpreting pictures of polyhedral scenes. *Artificial Intelligence*, 4(2):121–137, 1973.

[55] G. Markowsky and M. A. Wesley. Fleshing out wire frames. *IBM Journal of Research and Development*, 24(5):582–597, 1980.

[56] G. Markowsky and M. A. Wesley. Fleshing out projections. *IBM Journal of Research and Development*, 25(6):934–954, 1981.

[57] B. Marlin and G. Toussaint. Constructing convex 3-polytopes from two triangulations of a polygon. In *14th Canadian Conference on Computational Geometry*, pages 36–39, Lethbridge, Alberta, August 2002.

[58] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. Image-based visual hulls. In *SIGGRAPH 2000*, pages 369–374, New Orleans, Louisiana, July 2000.

[59] M. McKenna and R. Seidel. Finding the optimal shadows of a convex polytope. In *1st ACM Symposium on Computational Geometry*, pages 24–28, Baltimore, Maryland, June 1985.

[60] Melles Griot Corporation. *Machine Vision Guide*, 2003. `http://www.mellesgriot.com/products/machinevision/lif_3.htm`.

[61] J. Miller. Low-cost in-process machine vision gauging system. Technical report, Department of Electrical and Computer Engineering, University of Michigan-Dearborn, April 1998. `http://www.engin.umd.umich.edu/ceep/reports/96-97/jmiller.html`.

[62] J. A. Muratore. Illumination for machine vision. `http://www.pinnaclevision.co.uk/illum02.htm`.

[63] I. Nagendra and U. Gujar. 3-D objects from 2-D orthographic views– a survey. *Computer and Graphics*, 12(1):111–114, 1988.

[64] G. Nemhauser, A. R. Kan, and M. Todd, editors. *Optimization*, volume 1 of *Handbooks in Operations Research and Management Science.* Elsevier, Amsterdam, 1989.

[65] M. Penna. A shape from shading analysis for a single perspective image of a polyhedron. *IEEE Transaction of Pattern Analysis and Machine Intelligence*, 11(6):545–554, 1989.

[66] H. Plantinga and C. R. Dyer. Visibility, occlusion, and the aspect graph. *International Journal of Computer Vision*, 5(2):137–160, 1990.

[67] M. Pop, G. Barequet, C. A. Duncan, M. T. Goodrich, W. Huang, and S. Kumar. Efficient perspective-accurate silhouette computation and applications. In *17th ACM Symposium on Computational Geometry*, pages 60–68, Massachusetts, June 2001.

[68] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, New York, 1985.

[69] P. J. Ryan. *Euclidean and Non-Euclidean Geometry: An Analytic Approach.* Cambridge University Press, Cambridge, Massachusetts, 1986.

[70] R. D. Schiffenbauer. *A Survey of Aspect Graph.* PhD thesis, Department of Computer and Information Science, Polytechnic University, New York, 2001.

[71] W. B. Seales and C. R. Dyer. Viewpoint from occluding contour. *Computer Vision, Graphics and Image Processing: Image Understanding*, 1992.

[72] G. Shephard. Twenty problems on convex polyhedra—II. *Math. Gaz.*, 52:359–367, 1968.

[73] Siemens. *Outline inspection with SIMATIC VS 110.* Product literature. `http://www.ad.siemens.de/dipdata/mk/pdf/e20001-a60-p285-x-7600.pdf`.

[74] SIGHTech Vision Systems. *Eyebot Application, Inspecting Hard Disk Media.* Product literature. `http://www.sightech.com/hard_disk_app_note.pdf`.

[75] S. Skiena. *Geometric Probing.* PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, 1988.

[76] S. Skiena. Problems in geometric probing. *Algorithmica*, 7(4):599–605, 1989.

[77] S. Skiena. Geometric reconstruction problems. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry.* CRC Press, Boca Rotan, New York, 1997.

[78] J. Stolfi. *Oriented Projective Geometry: A Framework for Geometric Computations.* Academic Press, New York, 1991.

[79] K. Sugihara. A necessary and sufficient condition for a picture to represent a polyhedral scene. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 6(5):578–586, 1984.

[80] K. Sugihara. *Machine Interpretation of Line Drawing.* The MIT Press Series in Artificial Intelligence. MIT Press, Cambridge, Massachusetts, 1986.

[81] J. Taylor. Zonohedra and generalized zonohedra. *American Mathematical Monthly*, 99(2):108–111, 1992.

[82] G. Toussaint. The complexity of computing nice viewpoints of objects in space. In *Vision Geometry IX, Proc. SPIE International Symposium on Optical Science and Technology*, pages 1–11, San Diego, California, July 30–August 4 2000.

[83] P. A. C. Varley. *Automatic creation of boundary-representation models from single line drawings.* PhD thesis, Department of Computer Science, University of Wales College of Cardiff, 2003.

[84] W. Wang and G. G. Grinstein. Survey of 3d solid reconstruction from 2d projection line drawings. *Computer Graphics Forum*, 12(2):137–158, 1993.

[85] E. W. Weisstein. Latin square. From *Math World*– A Wolfram Web Resource, `http://www.mathworld.wolfram.com/LatinSquare.html`.

[86] Q.-W. Yan, C. L. P. Chen, and Z. Tang. Efficient algorithm for the reconstruction of 3d objects from orthograpgics projections. *Computer Aided Design*, 26(9):699–717, 1994.

[87] G. M. Ziegler. *Lectures on Polytopes.* Springer-Verlag, Berlin, 1995.