784

The Asp: A Continuous, Viewer-Centered Object Representation for Computer Vision

by

William Harry Plantinga

A thesis submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

(Computer Sciences)

at the

University of Wisconsin—Madison

1988

784

TABLE OF CONTENTS

ABSTRACT

In this thesis a new continuous, viewer-centered representation for polyhedral objects or scenes is introduced, called the *aspect representation* or *asp*. The asp is viewer-centered in the sense that it represents the appearance of a polyhedron to a viewer as a two-dimensional line drawing, rather than the volume of space that it fills. It is continuous in the sense that it represents appearance for all viewpoints rather than for a discrete set of viewpoints. In effect, it is a representation of appearance as a function of viewpoint. Analyses of the size of asps and algorithms for their construction are given under orthographic and perspective viewing models, for convex and non-convex polyhedra. The worst-case size of an asp is $\Theta(\mathbf{n})$ in the convex case and $\Theta(\mathbf{n}^4)$ in the non-convex case.

The asp is the first representation of appearance as a function of viewpoint. As such it is useful for problems that depend on knowing the appearance of an object from many viewpoints, finding ranges of viewpoints with particular characteristics of appearance, and determining viewpoints at which particular visual events happen. Many problems in computer vision and graphics fall into one of these categories. In general, the asp is useful for problems that make repeated use of appearance characteristics, justifying the representation of all visual events.

In this thesis the asp is applied to three problems, in each case yielding improvements or advantages over previous results. The first problem is the construction of the aspect graph, a graph defined to have a vertex for each topologically-distinct view of a polyhedron and edges connecting adjacent views. Using the asp, we present the first algorithm for constructing the aspect graph for general polyhedra. The second application is object recognition. In this application we show how to use the asp to represent and use occlusion information in recognizing three-dimensional objects from an arbitrary viewpoint. The third application is animating rotation, or showing views of a polyhedral scene from many closely-spaced viewpoints fast enough to give the appearance of the scene as the viewer moves around it.

Chapter 1: Representation and Appearance

1-1. Representation for Problem Solving

Any system that seeks to solve a problem must represent the problem to be solved; representation of some aspect of the world is one of the most basic requirements of a problem-solving system. Reasoning systems require a representation of knowledge and beliefs; natural language understanding systems require a representation of language and meaning; robot motion planning systems require a representation of location and shape of objects in the world.

A particular representation may be more or less helpful in the solution of a problem. Choosing an advantageous representation is often a key to finding an efficient solution because an advantageous representation makes explicit the information pertinent to the solution. For example, given two equivalent knowledge databases, one may yield a proof of a theorem much more quickly than the other if it includes key lemmas.

Problems in computer vision, computer graphics, CAD/CAM, robotics, and many other areas require a representation of objects in the world. Thus, it is not surprising that there are many different object representations in use with different advantages and disadvantages. Major existing three-dimensional (3-D) object representations can be divided into three main categories: volumetric, boundary, and swept-volume representations. Volumetric representations, such as octrees [Jackins & Tanimoto, 1980], space occupancy arrays [March & Steadman, 1974], [Srihari, 1981], and constructive solid geometry [Requicha, 1978], represent the volume of space that an object occupies by constructing the volume as a combination of simpler volumes. Boundary representations such as winged-edge polyhedra [Baumgart, 1972] represent the boundary of the volume. Swept-volume representations represent the cross-sectional area of the object swept along a spine [Binford, 1971]. However, all of the standard representations have something in common: they are object-centered in the sense that they represent the volume of Euclidean 3-space that the object occupies.

Object-centered representations have properties that make them useful for many problems. For example, the space taken to describe an object is generally small, and it is relatively easy to find the union and intersection of two objects. However, for many applications it is important to know the parts of a polyhedral scene that are visible. In computer graphics, it is necessary to calculate the parts of a scene that are visible in order to draw an image of the scene. In computer vision, it is important to know the parts of an object that are visible from various viewpoints in order to recognize the object in an image. In robot motion planning, the free space through which an object can move is bounded by what is visible in any direction.

Appearance is the pertinent information to problems such as these, but it is not represented explicitly in an object-centered representation. For such problems, it may be advantageous to explicitly represent the appearance of the object to a viewer from some or all viewpoints. We will call a representation of 3-D objects that represents appearance to a viewer rather than volume of space filled *viewer-centered* rather than *object-centered.*

1-2. Representing Appearance

Most of the effort in representing appearance to date has been in representing the appearance of an object from a single viewpoint. In computer-aided design, an object-centered representation of an object is constructed, and the appearance is computed and displayed from a single viewpoint or a couple of viewpoints at once. Flight simulators compute what is visible out of the cockpit window from each of a series of viewpoints. The central problem in computer graphics is computing the appearance of a scene from a given viewpoint.

In computer vision, there is increasing interest in representing appearance from several viewpoints for 3-D object recognition. In the "characteristic view" approach to 3-D object recognition, several views of an object from different, "commonly-occurring" viewpoints are stored in order to represent the object for recognition (e.g., [Rosenfeld, 1987]). The motivation is that since humans can recognize objects in around 100 neuron-firing times, it must be possible to recognize objects with very few computational steps. One conceivable way that recognition could occur so quickly is through comparisons of simple property measures on 2-D images.

Since these multiview representations represent appearance from a discrete set of viewpoints, they make it easy to determine properties of appearance from each of those viewpoints. However, there are infinitely many geometrically-distinct appearances of an object, and multiview representations must choose a representative few. In order to show the sufficiency of a set of views, it is necessary to define some criteria by which images and views match "closely enough" and use the definition to determine the views selected. The *visual potential graph* or *aspect graph,* introduced by Koenderink and van Doorn [1979], is useful to this end. The aspect graph has a vertex for each topologically-distinct view of an object. When a characteristic view of the object is stored for each vertex of the aspect graph, a *topological* match is guaranteed between an image of an object and some characteristic view. However, the match is still not a geometric one, and it is not obvious that a topological match is sufficient for recognition.

All of these representations are of appearance from a particular viewpoint or from a set of viewpoints. It would be useful to represent appearance from a continuous range of viewpoints or from all viewpoints. How can this be done? The answer lies in noting that appearance (or aspect) changes *continuously* with viewpoint, with occasional discontinuous changes in the structure of the image. We take advantage of this fact to represent the appearance of an object for all viewpoints, or as a function of viewpoint. We call this representation the *aspect representation* or *asp*.

1-3. The Asp: A Continuous, Viewer-centered Object Representation

When a viewer sees the world, she sees a 2-D projection of the visible part of the world, the image plane. As the viewer moves and the viewpoint changes, the appearance of the scene changes. Scene features usually change continuously with viewpoint. From some viewpoints there are changes in the structure of the image. The asp is a representation of appearance as a piecewise continuous function of viewpoint. It is continuous where the image changes continuously with viewpoint, and the continuous pieces are joined at visual events or topological changes in the appearance. Thus, the asp does not represent various individual two-dimensional views of the object; rather, it represents the volume of *aspect space* that the object occupies, where aspect space is a 4-D space consisting of two degrees of freedom in viewing direction (viewpoint space) and two degrees of freedom in appearance (image space) under our orthographic model.

In this thesis we define the asp and give algorithms for its construction under two viewing models, one using orthographic projection and a 2-D viewpoint space, and one using perspective projection and a 3-D viewpoint space. We present tight bounds on minimum and maximum size of the asp, for convex and non-convex polyhedra: the asp has $\Theta(\mathbf{n})$ size in the convex case and a worst-case size of $\Theta(\mathbf{n}^4)$ in the non-convex case. We also discuss other properties of the asp. Our model of computation allows for real arithmetic.

The aspect representation is useful for problems in which the appearance of objects is important to the solution. This may involve using appearance from many different viewpoints, finding ranges of viewpoints with particular appearance characteristics, or determining viewpoints at which particular visual events happen. We have studied three applications for which the asp is a beneficial representation: constructing the aspect graph, object recognition, and animating rotation.

The first application considered is the construction of the aspect graph. The aspect graph is a tool for characterizing information about the topological change of an image with viewpoint. It consists of a vertex for every topologically-distinct view of the object and edges connecting vertices corresponding to adjacent views. We use the asp to develop algorithms for constructing the aspect graph under the orthographic and perspective models. The aspect graph has maximum size $\Theta(\mathbf{n}^6)$ in the orthographic case and a maximum size of $\Theta(\mathbf{n}^9)$ in the perspective case. In both cases, using the asp yields the first algorithm for constructing the aspect graph for polyhedra. The algorithms usually require time within a factor of $O(\log \mathbf{m})$ of optimal in the sense that for most polyhedra, they require time $O(\mathbf{m} \log \mathbf{m})$ for an output size of $\mathbf{m}$.

The second application is object recognition. Marr argues that object-centered representations are best suited to object recognition, for reasons of parsimony and independence of viewpoint [Marr, 1982]. Rosenfeld argues that viewer-centered representations are probably necessary to recognize objects as quickly as humans can [Rosenfeld, 1987]. We show that it is possible to use the asp for object recognition to gain some of the benefits of both approaches. The asp requires less space than viewer-centered representations of all of the topologically-distinct views of an object. In addition, the asp represents appearance, rather than the volume of space filled, simplifying the recognition process, and it represents appearance from all viewpoints rather than a discrete set to achieve a sort of viewpoint independence. Thus any features of appearance that occur in an image, such as T-junctions, occlusions, and image properties, can be matched with a representation of appearance. The representation is particularly suited to an associative-memory or parallel architecture, with which all of the visibility information can be compared to the image at once.

The final application considered is animating the rotation of a polyhedral scene. That is, given a path of viewpoints in viewpoint space and a polyhedral scene, the problem is to efficiently display a series of images of the scene from closely-spaced viewpoints along the path, giving the appearance of a continuously rotating scene or moving viewer. We present an algorithm that takes advantage of the similarity from frame to frame by computing the initial appearance of the scene and the differences between frames. The algorithm makes use of the asp and the aspect graph constructed for the 1-D path of viewpoints. The algorithm uses the orthographic model and a linear path of viewpoints; extensions to the perspective model and to other paths of viewpoints are also discussed.

Chapter 2 of this thesis introduces the aspect representation, gives algorithms for its construction, and analyzes size and other properties. The asp is initially defined under an orthographic viewing model; the differences under a perspective model are also considered. Chapter 3 shows how to use the asp to construct the aspect graph. The aspect graph is constructed for convex and non-convex polyhedra, under the orthographic and perspective viewing models. Size bounds are also presented. Chapter 4 discusses the application of the asp to object recognition. It also compares asp object recognition with other common algorithms. Chapter 5 discusses using the asp to animate rotation and compares this method with other methods. Chapter 6 contains an annotated bibliography of related work and comparisons with our approach, and Chapter 7 contains conclusions and directions for further research.

Chapter 2: The Aspect Representation

2-1. Introduction

The aspect representation is a continuous, viewer-centered representation for polyhedra. It is viewer centered in the sense that it represents the appearance of a polyhedron to a viewer, rather than the volume of space that it fills, and it is continuous in the sense that it represents appearance as a piecewise continuous function of viewpoint. The asp represents the appearance of an object over all viewpoints, that is, image space for all viewpoints. Thus, one can think of it as representing the volume of *aspect space* that the object fills, where aspect space is image space × viewpoint space.

Section 2-2 introduces the aspect representation and discusses some of its properties under orthographic projection. Section 2-3 describes algorithms for the construction of the asp and for finding the cross-section of the asp at a given viewpoint. Section 2-4 contains a discussion of the differences in the asp when perspective projection is used. Section 2-5 contains concluding remarks and a discussion of the advantages and disadvantages of the aspect representation.

2-2. The Asp under Orthographic Projection

In order to define the aspect representation, we first define *viewpoints* and *viewpoint space*. In this section we use orthographic projection in constructing images from three-dimensional objects or scenes. We also assume that the whole object is in front of the viewer. Given these two assumptions, the distance from the camera to the scene is not significant, so we can define viewpoints as points on a unit sphere centered on the object. A vector from a point on the sphere (i.e. a viewpoint) to the center of the sphere is the viewing direction corresponding to that viewpoint. We speak of viewpoints, points on the sphere of viewpoints, and viewing directions interchangeably. Note that under orthographic projection one must think of sphere as being infinitely large if one wishes to think of the viewpoints as endpoints of lines of sight.

We define the viewpoint (θ,ϕ) to be the point on the unit sphere as shown in Figure 2-1. That is, if the point (0,0,1) corresponds to the viewing direction "straight ahead" (i.e. some reference viewing direction) then that point rotated by θ clockwise about the y-axis and then by ϕ clockwise about the x-axis corresponds to the viewing direction (θ,ϕ). (Figure 2-1 shows a negative ϕ and positive θ rotation.)

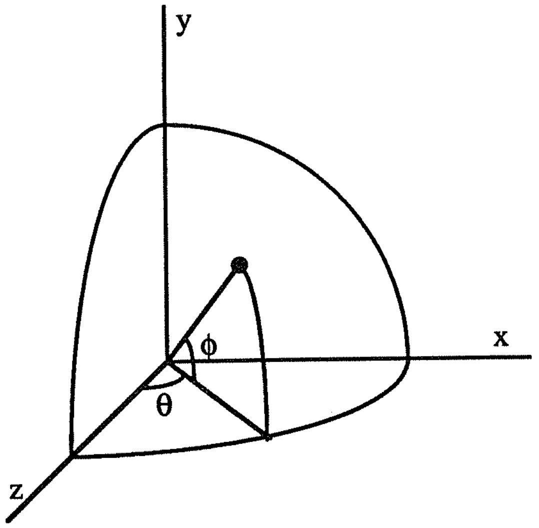


Figure 2-1. The viewpoint (θ,ϕ) on the unit sphere.

We consider the image of an object as a projection of the object onto a two-dimensional viewing plane, with hidden lines and surfaces removed. We call this viewing plane *image space* and denote a point in image space by (u,v). The image is assumed to have a fixed, "upright" orientation with respect to the object frame of reference. The aspect representation represents the image of the object as a function of viewpoint. Thus, it represents the volume of aspect space that the object occupies, where aspect space has an image plane for every viewpoint. One can think of it locally as image space × viewpoint space. In this definition the orientation of the image plane is singular at the poles, so we define the orientation at the poles to be that which would be reached as ϕ approaches ±90° with θ = 0°. One way to avoid this problem in practice is to treat the viewpoints at the poles as undefined, so that viewpoints can be arbitrarily close to, but not exactly on, either pole.

The asp for a face of a polyhedron is the volume of aspect space that the face occupies, that is, the object's appearance in the viewing plane for every viewpoint (θ,ϕ). Thus, the point (θ,ϕ,u,v) is in the asp for a face whenever (u,v) is in the image of that face from the viewpoint (θ,ϕ). The (θ,ϕ) cross-section of the asp for the face is then the appearance of the face from the viewpoint (θ,ϕ). The asp for a polyhedron is the union of the asps for its faces.

We construct the asp for a point at location (x_0,y_0,z_0) in a fixed coordinate system by considering what happens to the image of the point as the viewpoint changes. The projection onto the image plane can be separated into two parts: a rotation so that the viewing direction is along the z-axis and an orthographic projection into the z=0 plane. The rotation is given by

$$[x_0, y_0, z_0] \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} =$$

$$\begin{aligned} [&x_0 \cos\theta - z_0 \sin\theta, \\ &x_0 \sin\theta \sin\phi + y_0 \cos\phi + z_0 \cos\theta \sin\phi, \\ &x_0 \sin\theta \cos\phi - y_0 \sin\phi + z_0 \cos\theta \cos\phi] \end{aligned} \qquad (2\text{-}1)$$

After orthographic projection into the image plane (u,v), this yields

$$u = x_0 \cos\theta - z_0 \sin\theta \qquad (2\text{-}2)$$
$$v = x_0 \sin\theta \sin\phi + y_0 \cos\phi + z_0 \cos\theta \sin\phi \qquad (2\text{-}3)$$

These equations have two degrees of freedom, θ and ϕ. Thus a point in 3-space has as its aspect representation a 2-D surface in aspect space.

The asp for a line segment is a 3-D surface in aspect space or 3-surface, bounded by 2-surfaces. It can be written down directly by substituting parametric equations for a line segment into the point equations. We use a parametric representation for the line segment from (x_0,y_0,z_0) to (x_1,y_1,z_1), with parameter s varying from 0 to 1. Letting $a_1 = x_1-x_0$, $b_1 = y_1-y_0$, and $c_1 = z_1-z_0$ the parametric equations for a line segment are

$$\begin{aligned} x(s) &= x_1 + a_1 s \\ y(s) &= y_1 + b_1 s \\ z(s) &= z_1 + c_1 s \end{aligned} \qquad (2\text{-}4)$$

Substituting them into the point equations yields

$$u = (x_1 + a_1 s) \cos\theta - (z_1 + c_1 s) \sin\theta \qquad (2\text{-}5)$$
$$v = (x_1 + a_1 s) \sin\theta \sin\phi + (y_1 + b_1 s) \cos\phi + (z_1 + c_1 s) \cos\theta \sin\phi \qquad (2\text{-}6)$$

This is a 3-surface in aspect space. The bounding 2-surfaces are given by Eqs. (2-2) and (2-3) above for the points (x_0,y_0,z_0) and (x_1,y_1,z_1).

The asp for a triangle consists of the asps for the bounding sides and vertices as described above plus connectivity information, namely, links from the asps for bounding sides to the asps for the adjacent vertices, and so on. Figure 2-2 shows the triangle (1,1,0), (2,1,0), (1,2,0) in object space, and Figure 2-3 shows its asp. Since it is difficult to represent a 4-D volume in a 2-D figure, the figure shows two 3-D cross-sections of the asp at different fixed values of ϕ. The 2-D cross-sections of the asp within each 3-D cross-section are the images of the triangle for the corresponding values of θ and ϕ. The asp represents all of these cross-sections continuously by representing the equations of the bounding surfaces.

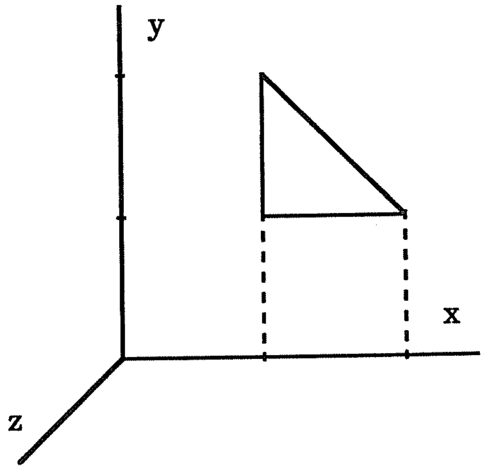


Figure 2-2. The triangle (1,1,0), (2,1,0), (1,2,0) in object space.

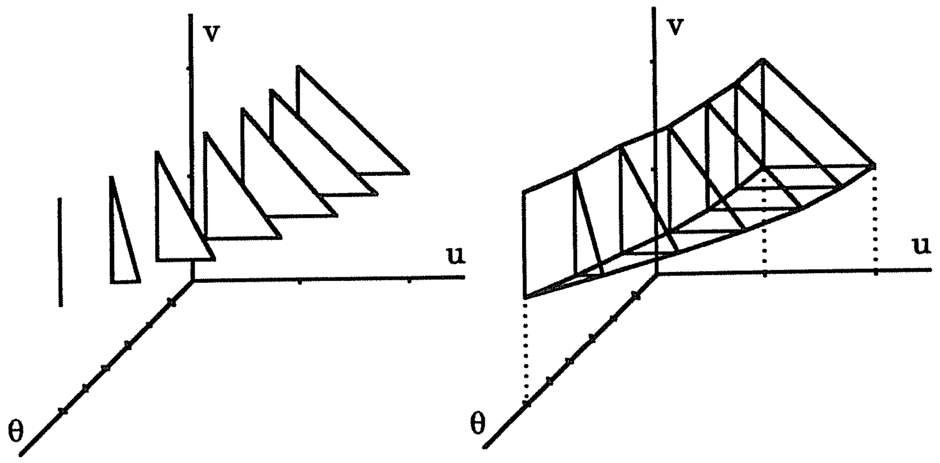


Figure 2-3a. A cross-section of the asp for the triangle for ϕ = 0°.

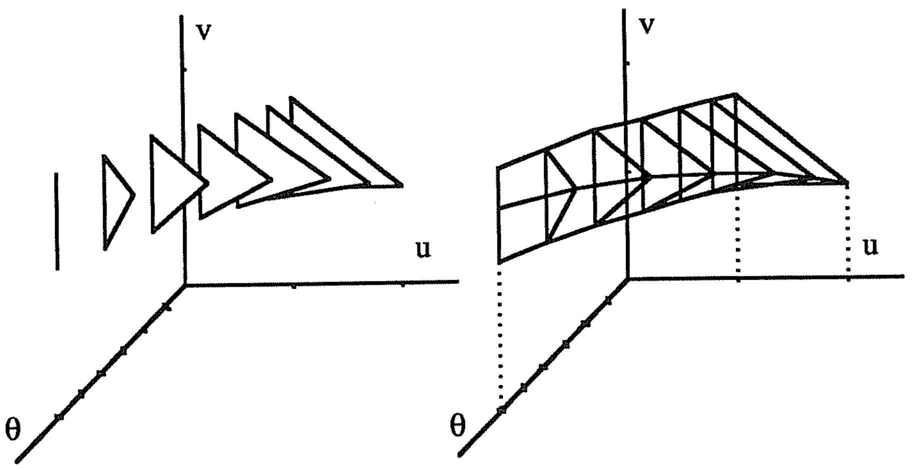


Figure 2-3b. A cross-section of the asp for the triangle for $\phi = 30^\circ$.

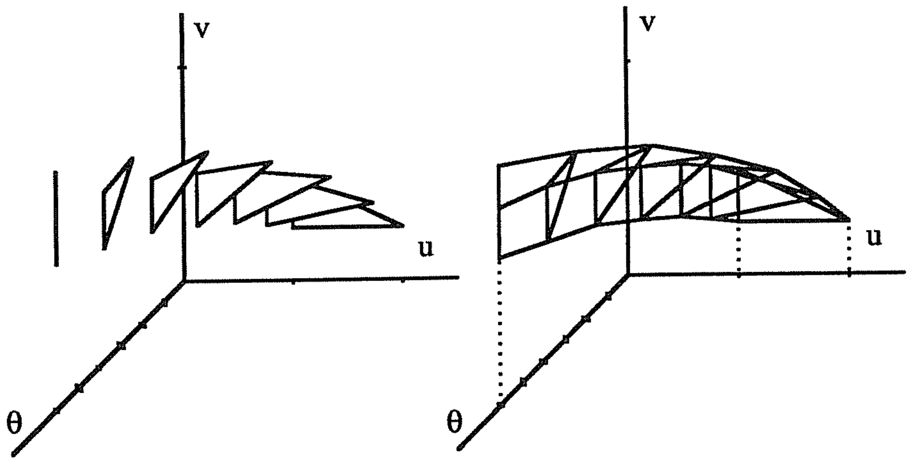


Figure 2-3c. A cross-section of the asp for the triangle for $\phi = 60^\circ$.

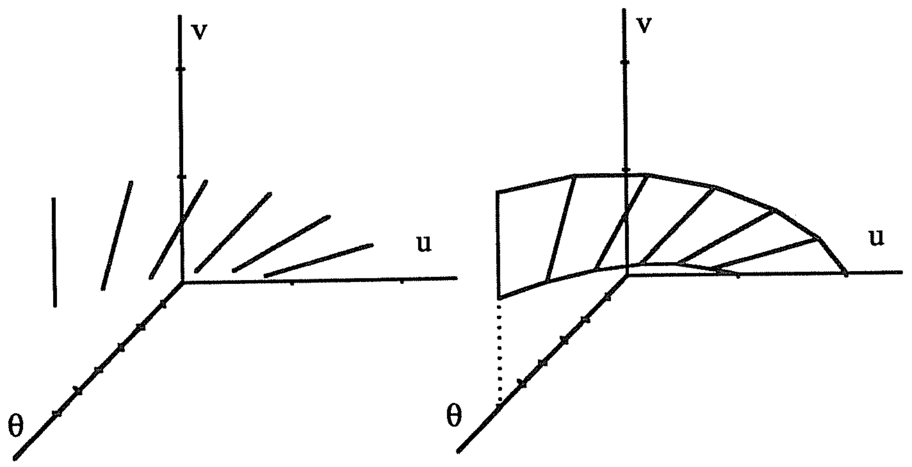


Figure 2-3d. A cross-section of the asp for the triangle for $\phi = 90^\circ$.

2-2.1. Representing Asps

We represent an asp by representing the volume of aspect space that it occupies, specifically, by representing the 3-surfaces that bound the 4-D cells of aspect space. These 3-surfaces are bounded by 2-surfaces, the 2-surfaces by 1-surfaces, and the 1-surfaces by points. Unfortunately, the surfaces are not in general planar, so the object is not a polytope (i.e. the generalization of a polygon, polyhedron, etc.). However, they are well-behaved: the trigonometric functions in the equations for the surfaces can be eliminated by a spherical-to-Cartesian coordinate transformation of the representation for viewpoints, from (θ,ϕ) to $\mathbf{V} = (v_x, v_y, v_z)$. As a result, the surfaces are algebraic, and each can be represented with a few constants.

Therefore we can work with the surfaces in much the same manner as we would work with lines and planes in a polyhedron. For example, we can calculate the intersection of the general form for any two of these surfaces in closed form. The intersection of two 3-surfaces is a 2-surface, a linear function of viewpoint. This kind of surface occurs in edge-vertex occlusion (Section 2-3.2.1.2). The intersection of three 3-surfaces is a 1-surface, a quadratic function of viewpoint. A vertex is the result of the intersection of four 3-surfaces; it is a cubic function of viewpoint. The equations for these surfaces are developed below.

Thus, while the asp is not a polytope, we speak of it as if it were. We refer to the 3-surfaces that bound the asp as *facets*, a term one would normally use for the 3-D faces bounding a 4-polytope. We refer to the 2-surfaces that bound the facets as *ridges* and the 1-surfaces or curves as *edges*. A 4-D volume of aspect space bounded by facets is referred to as a *cell*. In general, the asp is a 4-manifold in aspect space, i.e. a region or regions of aspect space partitioned into cells, which we refer to as a *subdivision*. The cells correspond to polygons in the image of the object, and the facets bounding the cells correspond to edges.

The asp is very much like a set of 4-polytopes in aspect space except that the "faces" are curved, and we must store the constants of the equation for each face. We represent the asp with a data structure similar to that of a polyhedron, except that the asp is 4-D rather than 3-D and the volume of the asp is partitioned into 4-D cells (see Figure 2-4).

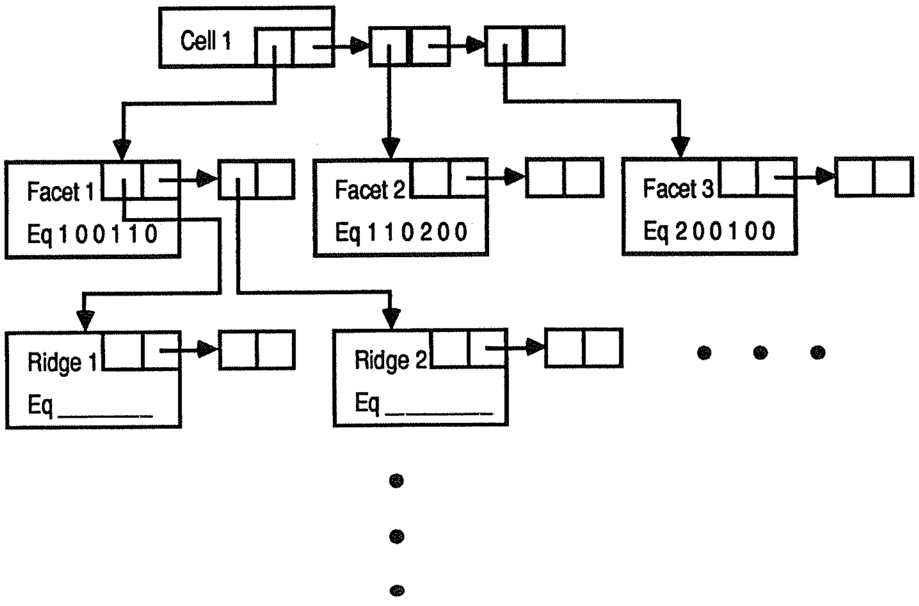


Figure 2-4. The data structure used for representing an asp.

A cell of aspect space is represented as a node with pointers to the bounding facets. The facets are represented by the constants of the equation for the 3-surface and pointers to the bounding ridges, and so on.

2-2.2. Asps and Visibility

Eqs. (2-5) and (2-6) for the asp for a point are defined for all values of θ and ϕ. This is as one would expect, since a point is visible from any viewing direction. We can represent a point which is not visible from every viewing direction, say a point that is occluded from some directions, by putting bounds on the values of θ and ϕ for Eqs. (2-5) and (2-6) above. That is, if we represent only a part of the surface for the point in aspect space, rather than the whole surface, we have the asp for a point which is visible from only some viewpoints.

In fact, a fundamental property of aspect space is that occlusion in object space corresponds to set subtraction in aspect space. That is, if one polygon partially occludes another, we can characterize that occlusion in aspect space by subtracting the asp of the first from the second. For example, Figure 2-5 shows two triangles in object space, one in the z=0 plane (left) and one in the z=1 plane; Figure 2-6 shows the asps for the triangles with $\phi = 0$.

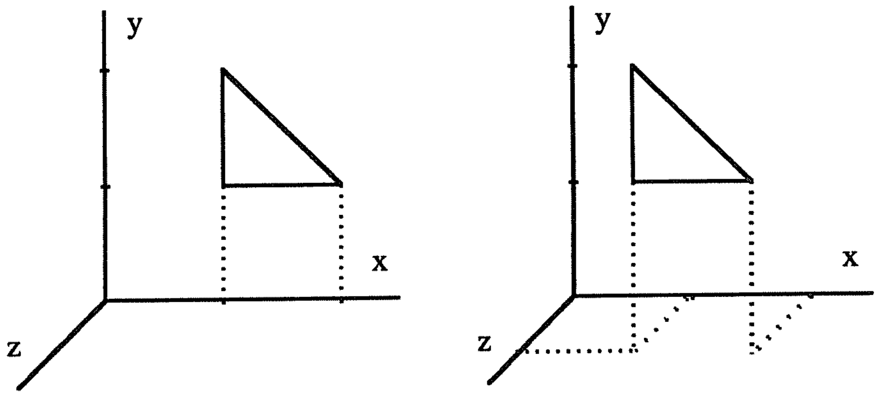


Figure 2-5. Two triangles in object space.

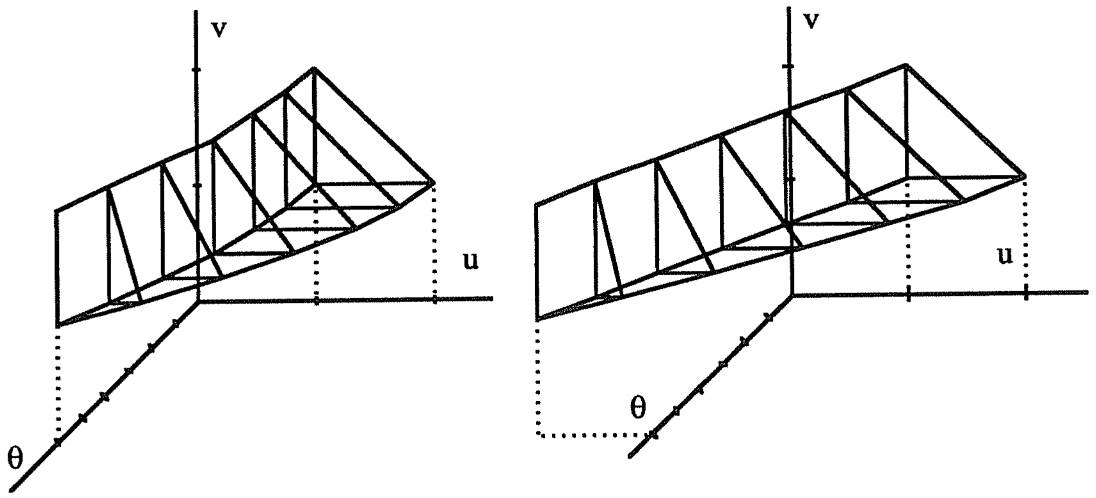


Figure 2-6. The asps for two triangles.

Figure 2-7 shows the set subtraction of the second triangle from the first. Since set subtraction in aspect space corresponds to occlusion in object space, any cross section of Figure 2-7 for a particular viewpoint corresponds to the part of the first triangle visible from that viewpoint and not occluded by the second.

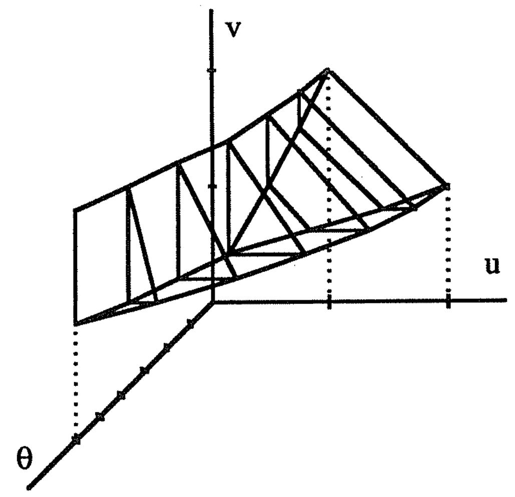


Figure 2-7. The asp for the left triangle as occluded by the right is the subtraction of the asp for the second from the asp for the first.

Note that if we take the projection of the asp for $\mathbf{p}_1$ as occluded by $\mathbf{p}_2$ onto viewpoint space, we get exactly the region of viewpoint space in which some point of $\mathbf{p}_1$ is visible. Thus, given a polyhedron or polyhedral scene, we can find the region of viewpoint space in which some face **f** of the polyhedron is visible by subtracting the asp for every other face from the asp for **f** and taking the projection of that asp onto (θ,ϕ).

Thus, occlusion in object space corresponds to set subtraction in aspect space. This is an important and interesting property of aspect space, which has many implications. For example, if we construct the aspect representation for a polyhedral scene, any cross-section of the asp from some viewpoint is an image of the scene from that viewpoint with hidden surfaces removed. In that sense we are pre-computing hidden-line removal information for a scene from all viewpoints.

Since the asp represents appearance, features of the appearance of an object, i.e. visual events, correspond directly to features of the asp. For example, an edge in an image is visible from a 2-D range of viewpoints and has 1-D extent in image space, so there is a corresponding 3-D surface in aspect space representing it. Two non-intersecting edges of an object can appear to intersect in a single point in a "T-junction" (see Figure 2-8) from a 2-D manifold of viewpoints, so there is a 1-face in the asp corresponding to that intersection point. In Figure 2-8, a 1-D cross-section of such a 2-D surface is shown as a bold curve.

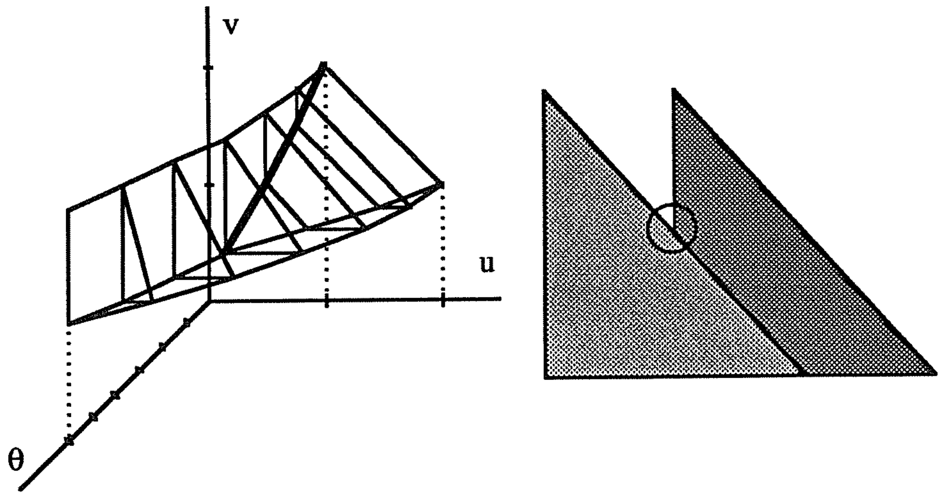


Figure 2-8. Representing features of occlusion: the T-junction on the right is represented in the asp by the bold curve on the left.

In order to find the feature of the asp corresponding to a visual feature, note what happens to the feature as the viewpoint varies. The dimension of the visual event in aspect space determines its representation in the asp. For example, a T-junction is a 0-D feature in the image, and it is visible from a 2-D range of viewpoints, so it is represented by a 2-D face of the asp. Three edges of an object can appear to intersect in a single image point from a 1-D curve of viewpoints, so the apparent intersection of three object edges in an image is a 1-D visual event, represented by a 1-D face of the asp. Table 2-1 presents a list of features in the appearance of an object and the corresponding features in the asp.

Visual Event	**Asp feature**
point in object space	ridge (2-face)
edge in object space	facet (3-face)
polygon in object space	cell (4-face)
apparent intersection point of two edges	ridge
apparent intersection of edge and vertex	edge (1-face)
apparent intersection of three edges	edge
apparent intersection of two vertices	vertex
apparent intersection of four edges	vertex

Table 2-1. Visual events and corresponding asp features.

2-2.3. Size of Asps

The size of an asp is defined as the amount of storage necessary to represent it. The space needed to represent an asp face is constant, so the size is proportional to the number of faces in the asp and the space needed to keep pointers between adjacent faces, i.e. the number of face-face adjacencies. In the absence of degeneracy, four asp edges meet in a vertex, three ridges in an edge, and so on, so the number of adjacencies is proportional to the number of faces and the number of faces is proportional to the number of asp vertices. Degeneracy serves only to reduce the number of faces. Thus, a bound on the number of asp vertices is a bound on the asymptotic size of the asp.

In the case of a convex polyhedron, the asp has size $O(\mathbf{n})$ for an object with $\mathbf{n}$ faces. This is true because the number of cells in the asp is equal to the number of faces of the polyhedron, the number of facets in the asp is twice the number of edges in the polyhedron, and the number of ridges of the asp is twice the number of facets. Each ridge of the asp corresponds to the intersection of two edges on a face, i.e. a vertex of the face. The face is visible from viewpoints in front the plane containing it, so there is one edge bounding each ridge for the face, which projects to a great circle in viewpoint space (see Section 2-3.1).

In the non-convex case the cross section of an asp at any value of (θ,ϕ) is a view of the corresponding object with hidden lines removed, so the size is clearly greater. In fact, the worst case size is $O(\mathbf{n}^4)$: each vertex arises as the apparent intersection of four object edges in an image. Since any set of four lines has a constant number of apparent intersection points, there are $O(\mathbf{n}^4)$ apparent intersection points of $\mathbf{n}$ lines.

We can find better bounds on the size of the asp given various restrictions on the object for which the asp is constructed. These restrictions are on the number of vertices of the asp, and as we showed above, they are also restrictions on the size of the whole asp in the absence of degeneracy. They are also restrictions on the size of the asp with degeneracy, if we count the "separate" vertices that have been collapsed into one by degeneracy. In other words, they are restrictions on the size of the asp if we count a vertex for every four object edges that appear to intersect in a point in an image.

These bounds can be determined by examining Table 2-1 above. In Table 2-1 we see that there are three different ways a vertex in the asp can occur: the apparent intersection of two vertices in the polyhedron at the same point in the image, the apparent intersection of a vertex and two unconnected edges, and the apparent intersection of four unconnected edges. Observe that the number of ways two different vertices in the object can appear at the same point is $O(\mathbf{n}^2)$, the number of ways a vertex and two edges can appear to intersect is $O(\mathbf{n}^3)$, and the number of ways four edges can appear to intersect is $O(\mathbf{n}^4)$. Thus, if the object is simple enough that the number of ways four edges appear to intersect in the same point is $O(\mathbf{n}^3)$, then we can immediately say that the size of the asp for that object is $O(\mathbf{n}^3)$. Also, if the number of ways two edges and a point appear to intersect in a point is $O(\mathbf{n}^2)$, then the size of the asp for that object is $O(\mathbf{n}^2)$. These bounds are summarized in Table 2-2.

Type of Polyhedron	Asp Size
general	$O(\mathbf{n}^4)$
$O(\mathbf{n}^3)$ ways that four edges appear to intersect in a single point (e.g. no grid behind a grid, but a picket fence behind a grid is acceptable)	$O(\mathbf{n}^3)$
$O(\mathbf{n}^2)$ ways that four edges appear to intersect in a single point (e.g. no picket fence behind a grid, but a grid is acceptable)	$O(\mathbf{n}^2)$
convex	$O(\mathbf{n})$

Table 2-2. Bounds on the size of the asp.

These restrictions are not unrealistic. For example, the only way to get an asp of size $\Omega(\mathbf{n}^4)$ is to have that many cases where four edges overlap in an image. An example of such a situation is a pair of grids, one behind the other (see Figure 2-9).

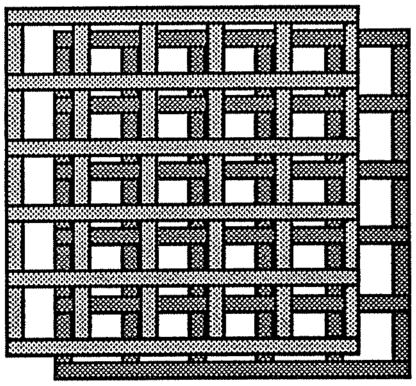


Figure 2-9. An example of an object for which the asp has size $\Theta(\mathbf{n}^4)$.

The example in Figure 2-9 shows that $\Theta(\mathbf{n}^4)$ is also a lower bound on the maximum size of the asp. The object consists of two grids, one behind the other, each of which consists of **n** strips laid in one direction and **n** in an orthogonal direction. We can choose an appropriate viewpoint so that any two vertical strips and two horizontal strips from different grids can have edges that appear to intersect in a single point. Thus, the asp for that object has $\Omega(\mathbf{n}^4)$ vertices.

This sort of situation doesn't occur often in natural scenes. Note also that in images with many faces, the reason for the large number of faces is often that they are approximating some sort of smooth surface. In this case, the complexity of the asp is much less than the worst case, since large numbers of faces are on convex surfaces and the number of ways that four object edges can appear to intersect in the image is relatively small.

2-3. Algorithms for Asps

Earlier it was shown that occlusion in object space corresponds to subtraction in aspect space. That is, the visibility of a face **f** as occluded by another face in front of it is the subtraction of the asp for the second face from the asp for **f**. Therefore, the asp for a face **f** as occluded by several other faces is the subtraction of the asps for the faces in front of the plane containing **f** from the asp for **f**. If a face is partially in front of **f** and partially behind it, then only the asp for the part in front of **f** should be subtracted.

Thus, to construct the asp for a face of a polyhedron we subtract the asps for the faces or parts of faces in front of it from the asp for that face. The asp for the whole polyhedron is the union of the asps for its faces. We will show how to find the subtraction of one volume in aspect space from another using an algorithm analogous to a common algorithm for polyhedra. To find the intersection of two polyhedra, it suffices to find the intersection of every pair of faces and join the results appropriately. Similarly, we find the intersection of two asps by finding the intersections of every pair of facets and joining appropriately. Finding the intersection of pairs of facets involves finding the intersection of lower-dimensional faces.

The algorithm for asps is more complicated than the algorithm for polyhedra for two reasons. First, asps are volumes of a 4-D space, rather than 3-D, and second, the surfaces are curved, and we must be able to find the intersection of such curved surfaces. We represent the surfaces by storing constants to known equations, and intersections of pairs of surfaces are given by other equations.

The following section describes how to construct the asp for a single, unoccluded polygon, which is the first step in constructing the asp for a polyhedron. The next section gives the algorithm for constructing the asp for a polyhedral scene. An algorithm is also given for finding a cross-section of an asp at a given viewpoint, which is the appearance of the polyhedron from that viewpoint with hidden lines removed.

2-3.1. Constructing the Asp for a Single Polygon

The asp for an unoccluded polygon is a 4-D cell of aspect space, consisting of the points (θ,ϕ,u,v) such that the polygon is visible at image point (u,v) from viewpoint (θ,ϕ). We will call the faces of the asp *vertices*, *edges*, *ridges*, *facets*, and *cells*, according to whether they are 0-, 1-, 2-, 3-, or 4-dimensional. The cell is bounded by facets, the facets by ridges, and so on. The facets of the asp correspond to the polygon edges, and the ridges correspond to the polygon vertices.

The asp edges and vertices do not correspond to any single part of the polygon; rather, they correspond to visual events bounding the visibility of the polygon. Since the polygon is unoccluded, it is visible from all viewpoints in front of the plane containing it. Thus, the polygon is visible from viewpoints in one hemisphere of viewpoint space and not visible from the other viewpoints, and the visual event boundary is a great circle of viewpoints.

A ridge in the asp corresponds to a vertex of a face, and since the face is visible from a hemisphere of viewpoints, the vertex as a boundary of the face is also visible from the same hemisphere of viewpoints. Thus, the asp edges and vertices that form the boundary for a ridge will be a great circle in a plane parallel to the face. The great circle can be represented as a single asp edge without any bounding vertices, but it reduces the number of special cases in the algorithms below to represent the great circle as a pair of asp edges between two arbitrary points on the circle. The actual representation used should be consistent with that described below of the asp for a polyhedral scene.

2-3.2. Constructing the Asp for a Polyhedron or Polyhedral Scene

The asp for a polyhedron or polyhedral scene is the union of the cells of aspect space in which each face of the polyhedron is visible. However, because faces can be partially occluded by other faces, constructing the asp for a face of a polyhedron is more complicated than for an unoccluded polygon. The asp for a face **f** of a polyhedron as occluded by the other faces is the asp for **f** minus (in the sense of set subtraction) the asps for the faces or parts of faces in front of **f**.

The subtraction of one asp from another is equivalent to the intersection of one asp with the complement of the other. Taking the intersection of two asps will introduce new kinds of faces to the asp. For example, in the case of an unoccluded polygon, a ridge of the asp corresponds to a vertex. Another kind of ridge that occurs for polyhedra corresponds to the *apparent* intersection of two edges of the polyhedron. That is, two edges that in fact do not intersect may appear to intersect from some viewpoints if one is in front of the other. This visual event is visible from a 2-D range of viewpoints and has 0-D extent in image space, so it is a ridge of the asp. It arises from taking the intersection of two asp facets.

Thus constructing the asp in the non-convex case requires calculating and representing additional kinds of asp ridges, edges, and vertices. These new kinds of faces arise as the intersection of asp surfaces. In Section 2-3.2.1 we derive standard forms for these surfaces and show how to find the intersection of pairs of asp surfaces. In Section 2-3.2.2 we show how to find intersections of asp cells. In Section 2-3.2.3 we describe in more detail how to construct the asp for a polyhedron.

2-3.2.1. Aspect Surfaces

The aspect representation for an unoccluded polygon is a cell of aspect space—a 4-D volume bounded by facets. The asp for a polygon that is a face of a non-convex polyhedron is also a cell of aspect space, bounded by facets. The facets again correspond to the edges bounding the polygon. Ridges bound the visibility of facets, and there are ridges corresponding to the vertices bounding the edges of the polygon. However, there is another kind of ridge, corresponding to the case where the visibility of a polygon edge is bounded by an occluding edge, i.e. a T-junction. In that case there is a ridge corresponding to the apparent intersection of the two edges. The surface on which the ridge lies results from the intersection of the surfaces on which the two facets lie. It is the general sort of 2-surface. We derive equations for the general 2-surface in Section 2-3.2.1.2.

The visual event that occurs in the case of an unoccluded polygon (the horizon effect) also bounds the visibility of a ridge in the non-convex case. However, another kind of visual event that bounds the visibility of a ridge in the non-convex case is the apparent intersection of three unconnected object edges in a single image point. We derive the general equations for such a boundary in Section 2-3.2.1.3.

Asp vertices correspond to visual events that are visible from a single viewpoint and that occupy a single point in the image plane. In the case of a non-convex polyhedron, the most general visual event is the apparent intersection of four object edges in a single point from some viewpoint. We derive the equations for such a point in Section 2-3.2.1.4.

In order to find the intersection of cells of aspect space, we must be able to find the intersection of pairs of asp surfaces of various kinds. Note that all of the asp surfaces—3-surfaces, 2-surfaces, 1-surfaces, and vertices—result from the intersection of a set of asp 3-surfaces—one, two, three, and four, respectively. Thus, finding the intersection of an asp 3-surface and an asp 2-surface is equivalent to finding the intersection of the three "parent" asp 3-surfaces involved. Also, since two asp 1-surfaces that lie on the same 2-surface have two common 3-surface "parents," finding the intersection of two 1-surfaces is equivalent to finding the intersection of the four unique 3-surfaces involved in the problem. Finding the intersection of two asp surfaces is equivalent to finding the intersection of the unique "parents" of those surfaces.

In order to find the intersection of asp cells, we must also be able to determine the orientation of one asp edge with respect to another another asp edge. To do so we will want to be able to compute a tangent to an asp edge at a point and a normal to that tangent. We show how to do this in Section 2-3.2.1.5.

2-3.2.1.1. 3-Surfaces of Aspect Space

A 3-surface of aspect space corresponds to the visibility of a line in object space. The equations for such a surface for a line passing through the points $\mathbf{p}_1 = (x_1,y_1,z_1)$ and $\mathbf{p}_1 + \mathbf{a}_1 = (x_1+a_1,y_1+b_1,z_1+c_1)$ were derived in Section 2-2 as:

$$u = (x_1 + s\, a_1) \cos\theta - (z_1 + s\, c_1) \sin\theta \qquad (2\text{-}7)$$

$$v = (x_1 + s\, a_1) \sin\theta \sin\phi + (y_1 + s\, b_1) \cos\phi + (z_1 + s\, c_1) \cos\theta \sin\phi \tag{2-8}$$

2-3.2.1.2. 2-Surfaces of Aspect Space

2-surfaces of aspect space arise in the general case as the intersection of two 3-surfaces. This may occur in two ways. As in the convex case, a 2-surface may correspond to a vertex of the object. In that case the 2-surface on which it lies is the intersection of the 3-surfaces corresponding to the object edges that meet at that vertex. The other way that an asp 2-surface can arise is as the intersection of two general 3-surfaces. In that case the resulting 2-surface corresponds to the visual event of the *apparent* intersection of two object edges, i.e. the intersection of two edges in an image, although the edges do not intersect in space. We derive the equations for the latter, more general kind of 2-surface. If the two object edges intersect in a point, the equations derived will simplify to the equations for the simpler case.

Suppose we have two 3-surfaces of the form of Eqs. (2-7) and (2-8) above, the second with subscripts of 2. The intersection of the two 3-surfaces is a 2-surface, which can be found by solving the two equations of the form of Eq. (2-7) for θ and of Eq. (2-8) for ϕ, yielding

$$\tan\theta = \frac{(x_2 + s_2\, a_2) - (x_1 + s\, a_1)}{(z_2 + s_2\, c_2) - (z_1 + s\, c_1)} \tag{2-9}$$

$$\tan\phi = \frac{-\,[(y_2 + s_2\, b_2) - (y_1 + s\, b_1)] \cos\theta}{(z_2 + s_2\, c_2) - (z_1 + s\, c_1)} \tag{2-10}$$

Since this is a 2-surface it depends only on two parameters. In Eqs. (2-9) and (2-10) the surface is given as a function of s and s_2. More often, however, we will want u and v as a function of θ and ϕ. Thus, we will solve (2-9) and (2-10) for viewpoint.

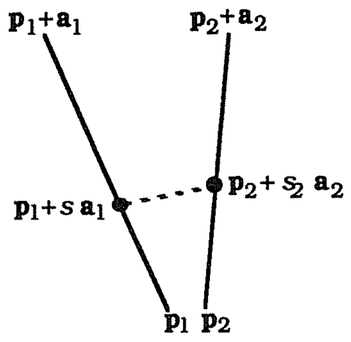
We can simplify the equations by eliminating sinusoidal functions through a change of variables. Instead of θ and ϕ, we can express the equations in terms of a viewpoint $\mathbf{v} = (v_x, v_y, v_z) = (\sin\theta \cos\phi, -\sin\phi, \cos\theta \cos\phi)$, so that the vector from **v** to the origin is the viewing direction (θ,ϕ). In other words, we substitute for (θ,ϕ) the unit vector **v** in the direction (θ,ϕ). Note that $|\mathbf{v}|=1$. For Eq. (2-9) we get

$$\frac{v_x}{v_z} = \frac{(x_2 + s_2\, a_2) - (x_1 + s\, a_1)}{(z_2 + s_2\, c_2) - (z_1 + s\, c_1)} \tag{2-11}$$

and for (2-10) we get

$$\frac{v_y}{v_z} = \frac{(y_2 + s_2\, b_2) - (y_1 + s\, b_1)}{(z_2 + s_2\, c_2) - (z_1 + s\, c_1)} \tag{2-12}$$

Note that we could have arrived at these equations immediately by noting that the viewing direction **v** must be parallel to a vector from one of the lines to the other at the points where they appear to intersect, i.e. a vector from $\mathbf{p}_1 + s\,\mathbf{a}_1$ to $\mathbf{p}_2 + s_2\,\mathbf{a}_2$ (see Figure 2-10).



$\mathbf{p}_1+\mathbf{a}_1$
$\mathbf{p}_2+\mathbf{a}_2$
$\mathbf{p}_2+s_2\,\mathbf{a}_2$
$\mathbf{p}_1+s\,\mathbf{a}_1$
$\mathbf{p}_1$
$\mathbf{p}_2$

Figure 2-10. The viewing direction along which two lines appear to intersect.

Solving (2-11) and (2-12) for s_2 yields

$$s_2 = \frac{v_x\,(z_1 - z_2 + s\,c_1) - v_z\,(x_1 - x_2 + s\,a_1)}{v_x\,c_2 - v_z\,a_2} \tag{2-13}$$

$$s_2 = \frac{v_y\,(z_1 - z_2 + s\,c_1) - v_z\,(y_1 - y_2 + s\,b_1)}{v_y\,c_2 - v_z\,b_2} \tag{2-14}$$

Solving (2-13) and (2-14) for s yields

$$s = \frac{\mathbf{v} \cdot [(\mathbf{p}_2 - \mathbf{p}_1) \times \mathbf{a}_2]}{\mathbf{v} \cdot (\mathbf{a}_1 \times \mathbf{a}_2)} \tag{2-15}$$

Eq. (2-15) can be substituted into Eq. (2-7) and (2-8) to find u and v.

The surface can be represented by four vectors or twelve constants: $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{a}_1$, and $\mathbf{a}_2$. These are related to the four endpoints of the two lines involved in the visual event. Thus, Eqs. (2-7), (2-8), and (2-15) represent a 2-surface in aspect space that results from the intersection of two 3-surfaces corresponding to asp facets.

2-3.2.1.3. 1-Surfaces of Aspect Space

The intersection of three 3-surfaces of aspect space is a 1-surface or curve. It corresponds to the visual event of the apparent intersection of three object edges in an image. Three object edges can appear to intersect from a 1-dimensional curve of viewpoints, and their apparent intersection is a single point, so the aspect representation for this visual event is a 1-surface or curve in aspect space.

The intersection of a 3-surface and a 2-surface is also a 1-surface, but this case is equivalent to the former case since the 2-surface is the intersection of two 3-surfaces. Another equivalent intersection problem is the intersection of two 2-surfaces, both of which lie on the same 3-surface, since in that case both 2-surfaces have a 3-surface "parent" in common.

We could find the 1-surface by finding the intersection of three pairs of equations of the form of Eqs. (2-7) and (2-8) for three 3-surfaces. However, there is a simpler method. Consider three edges in space, $\mathbf{p_1}$ to $\mathbf{p_1} + \mathbf{a_1}$, $\mathbf{p_2}$ to $\mathbf{p_2} + \mathbf{a_2}$, and $\mathbf{p_3}$ to $\mathbf{p_3} + \mathbf{a_3}$. Pick a point $\mathbf{p_1} + s\,\mathbf{a_1}$ on one line (see Figure 2-11) and find a line through that point that intersects both of the other lines.

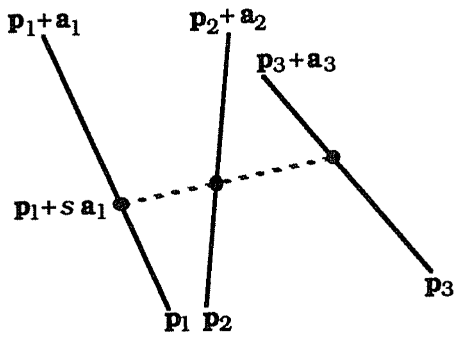

$\mathbf{p_1}+\mathbf{a_1}$
$\mathbf{p_2}+\mathbf{a_2}$
$\mathbf{p_3}+\mathbf{a_3}$
$\mathbf{p_1}+s\,\mathbf{a_1}$
$\mathbf{p_3}$
$\mathbf{p_1}$ $\mathbf{p_2}$

Figure 2-11. A viewing direction along which three object edges appear to intersect in a single point in an image.

Consider the planes defined by the point $\mathbf{p_1} + s\,\mathbf{a_1}$ and each of the other two edges. The viewing direction is parallel to the intersection of these two planes. A normal to the plane defined by the points $\mathbf{p_1} + s\,\mathbf{a_1}$, $\mathbf{p_2}$, and $\mathbf{p_2} + \mathbf{a_2}$ is given by

$$(\mathbf{p_1} + s\,\mathbf{a_1} - \mathbf{p_2}) \times \mathbf{a_2}$$

and a normal to the plane defined by the points $\mathbf{p_1} + s\,\mathbf{a_1}$, $\mathbf{p_3}$, and $\mathbf{p_3} + \mathbf{a_3}$ is given by

$$(\mathbf{p_1} + s\,\mathbf{a_1} - \mathbf{p_3}) \times \mathbf{a_3}$$

Therefore a vector parallel to the viewing direction in question (though not a unit vector) is given by

$$\mathbf{v}' = ((\mathbf{p_1} + s\,\mathbf{a_1} - \mathbf{p_2}) \times \mathbf{a_2}) \times ((\mathbf{p_1} + s\,\mathbf{a_1} - \mathbf{p_3}) \times \mathbf{a_3}) \qquad (2\text{-}16)$$

Thus

$$\mathbf{v}' = \langle s\,\mathrm{bc}_{12} + \mathrm{bz}_{12}, s\,\mathrm{ca}_{12} + \mathrm{cx}_{12}, s\,\mathrm{ab}_{12} + \mathrm{ay}_{12}\rangle \times \\ \langle s\,\mathrm{bc}_{13} + \mathrm{bz}_{13}, s\,\mathrm{ca}_{13} + \mathrm{cx}_{13}, s\,\mathrm{ab}_{13} + \mathrm{ay}_{13}\rangle$$

where $\mathrm{bc}_{12} = \mathrm{b}_1\,\mathrm{c}_2 - \mathrm{b}_2\,\mathrm{c}_1$, $\mathrm{bz}_{12} = \mathrm{c}_2\,\mathrm{y}_{12} - \mathrm{b}_2\,\mathrm{z}_{12}$, and so on. Therefore, if $\mathbf{v}' = \langle \mathrm{v_x}', \mathrm{v_y}', \mathrm{v_z}' \rangle$ we have

$$\begin{aligned} v_x' = {} & s^2 (ca_{12}\, ab_{13} - ab_{12}\, ca_{13}) \\ & + s (ca_{12}\, ay_{13} - ab_{12}\, cx_{13} + cx_{12}\, ab_{13} - ay_{12}\, ca_{13}) \\ & + (cx_{12}\, ay_{13} - ay_{12}\, cx_{13}) \end{aligned} \tag{2-17}$$

$$\begin{aligned} v_y' = {} & s^2 (ab_{12}\, bc_{13} - bc_{12}\, ab_{13}) \\ & + s (ab_{12}\, bz_{13} - bc_{12}\, ay_{13} + ay_{12}\, bc_{13} - bz_{12}\, ab_{13}) \\ & + (ay_{12}\, bz_{13} - bz_{12}\, ay_{13}) \end{aligned} \tag{2-18}$$

$$\begin{aligned} v_z' = {} & s^2 (bc_{12}\, ca_{12} - ca_{12}\, bc_{13}) \\ & + s (bc_{12}\, cx_{13} - ca_{12}\, bz_{13} + bz_{12}\, ca_{13} - cx_{12}\, bc_{13}) \\ & + (bz_{12}\, cx_{13} - cx_{12}\, bz_{13}) \end{aligned} \tag{2-19}$$

Since $\tan\theta = v_x/v_z = v_x'/v_z'$, we can substitute

$$\begin{aligned} e_1 &= ca_{12}\, ab_{13} - ab_{12}\, ca_{13} \\ e_2 &= ca_{12}\, ay_{13} - ab_{12}\, cx_{13} + cx_{12}\, ab_{13} - ay_{12}\, ca_{13} \\ e_3 &= cx_{12}\, ay_{13} - ay_{12}\, cx_{13} \\ e_4 &= bc_{12}\, ca_{13} - ca_{12}\, bc_{13} \\ e_5 &= bc_{12}\, cx_{13} - ca_{12}\, bz_{13} + bz_{12}\, ca_{13} - cx_{12}\, bc_{13} \\ e_6 &= bz_{12}\, cx_{13} - cx_{12}\, bz_{13} \end{aligned}$$

into Eqs. (2-17) and (2-19), yielding

$$\tan\theta = \frac{e_1\, s^2 + e_2\, s + e3}{e_4\, s^2 + e_5\, s + e_6} \tag{2-20}$$

Solving (2-20) for s in terms of θ yields

$$s^2 (e_4 \tan\theta - e_1) + s (e_5 \tan\theta - e_2) + e_6 \tan\theta - e_3 = 0 \tag{2-21}$$

Also, expressing ϕ in terms of θ and s yields

$$\tan\phi = \frac{-\,v_y' \cos\theta}{v_z'} \tag{2-22}$$

Thus, given some θ, we get s from Eq. (2-21) with the quadratic equation, v_x' and v_y' from Eqs. (2-17) and (2-18), and ϕ from Eq. (2-22). We can then get u and v from Eqs. (2-7) and (2-8). Therefore, Eqs. (2-7), (2-8), (2-17), (2-18), (2-21), and (2-22) represent an aspect 1-surface by giving u, v, and ϕ in terms of a quadratic function of θ.

2-3.2.1.4. Vertex of Aspect Space

A vertex of aspect space results from the intersection of four asp 3-surfaces, two general 2-surfaces, two 1-surfaces on a 2-surface, or any other combination of surfaces with four unique 3-surface "parents." The visual event that gives rise to an asp vertex is the apparent intersection of four object edges in a single point. This occurs from only one viewpoint, so the corresponding asp surface is a vertex.

We could find the vertex of aspect space corresponding to the apparent intersection of four object edges by finding the intersection of four pairs of equations of the form of Eqs. (2-7) and (2-8) for four 3-surfaces. However, there is a simpler method. Consider four lines in space, through $\mathbf{p}_1$ and $\mathbf{p}_1 + \mathbf{a}_1$, $\mathbf{p}_2$ and $\mathbf{p}_2 + \mathbf{a}_2$, $\mathbf{p}_3$ and $\mathbf{p}_3 + \mathbf{a}_3$, and $\mathbf{p}_4$ and $\mathbf{p}_4 + \mathbf{a}_4$. Pick a point $\mathbf{p}_1 + s\,\mathbf{a}_1$ on one line (see Figure 2-12).

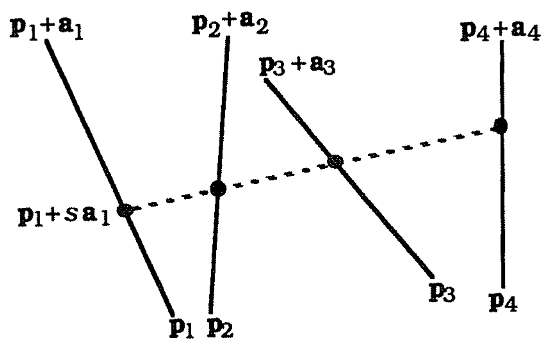


Figure 2-12. A viewing direction along which four object edges appear to intersect in a single point in an image.

A viewing direction through the point $\mathbf{p}_1 + s\,\mathbf{a}_1$ from which the four edges appear to intersect in a point must be parallel to the three planes defined by $\mathbf{p}_1 + s\,\mathbf{a}_1$ and each of the other lines. Therefore if there is such a line of sight, the triple cross product of the normals to the three planes must be zero. The normals to the planes are given by the cross product of two vectors in each plane:

$$(\mathbf{p}_1 + s\,\mathbf{a}_1 - \mathbf{p}_2) \times \mathbf{a}_2$$
$$(\mathbf{p}_1 + s\,\mathbf{a}_1 - \mathbf{p}_3) \times \mathbf{a}_3$$
$$(\mathbf{p}_1 + s\,\mathbf{a}_1 - \mathbf{p}_4) \times \mathbf{a}_4$$

If the three planes intersect in a single line, then the normals have a triple cross product of zero, which in determinant notation is

$$\begin{vmatrix} s\,\mathrm{bc}_{12}+\mathrm{bz}_{12} & s\,\mathrm{ca}_{12}+\mathrm{cx}_{12} & s\,\mathrm{ab}_{12}+\mathrm{ay}_{12} \\ s\,\mathrm{bc}_{13}+\mathrm{bz}_{13} & s\,\mathrm{ca}_{13}+\mathrm{cx}_{13} & s\,\mathrm{ab}_{13}+\mathrm{ay}_{13} \\ s\,\mathrm{bc}_{14}+\mathrm{bz}_{14} & s\,\mathrm{ca}_{14}+\mathrm{cx}_{14} & s\,\mathrm{ab}_{14}+\mathrm{ay}_{14} \end{vmatrix} = 0 \qquad (2\text{-}23)$$

In order to find the point of aspect space we must write out Eq. (2-23) in terms of the powers of s. We can then solve for s using the cubic equation.

For the s^3 term, only the bc_{ij}, ca_{ij}, and ab_{ij} terms of Eq. (2-23) are involved, so letting a be the coefficient of the s^3 term we get

$$a = \begin{vmatrix} bc_{12} & ca_{12} & ab_{12} \\ bc_{13} & ca_{13} & ab_{13} \\ bc_{14} & ca_{14} & ab_{14} \end{vmatrix} \tag{2-24}$$

For the s^2 term, we have one row of terms not involving s in each coefficient matrix, so we get

$$b = \begin{vmatrix} bz_{12} & cx_{12} & ay_{12} \\ bc_{13} & ca_{13} & ab_{13} \\ bc_{14} & ca_{14} & ab_{14} \end{vmatrix} + \begin{vmatrix} bc_{12} & ca_{12} & ab_{12} \\ bz_{13} & cx_{13} & ay_{13} \\ bc_{14} & ca_{14} & ab_{14} \end{vmatrix} + \begin{vmatrix} bc_{12} & ca_{12} & ab_{12} \\ bc_{13} & ca_{13} & ab_{13} \\ bz_{14} & cx_{14} & ay_{14} \end{vmatrix} \tag{2-25}$$

For the s term we have two rows of terms not involving s and one row involving s in each coefficient matrix:

$$c = \begin{vmatrix} bc_{12} & ca_{12} & ab_{12} \\ bz_{13} & cx_{13} & ay_{13} \\ bz_{14} & cx_{14} & ay_{14} \end{vmatrix} + \begin{vmatrix} bz_{12} & cx_{12} & ay_{12} \\ bc_{13} & ca_{13} & ab_{13} \\ bz_{14} & cx_{14} & ay_{14} \end{vmatrix} + \begin{vmatrix} bz_{12} & cx_{12} & ay_{12} \\ bz_{13} & cx_{13} & ay_{13} \\ bc_{14} & ca_{14} & ab_{14} \end{vmatrix} \tag{2-26}$$

Finally, the term that is constant in s involves only the bz_{ij}, cx_{ij}, and ay_{ij} constants:

$$d = \begin{vmatrix} bz_{12} & cx_{12} & ay_{12} \\ bz_{13} & cx_{13} & ay_{13} \\ bz_{14} & cx_{14} & ay_{14} \end{vmatrix} \tag{2-27}$$

Then we can solve the cubic equation

$$a\, s^3 + b\, s^2 + c\, s + d = 0 \tag{2-28}$$

for s. Once we have found s, we can find v_x', v_y', and v_z' using Eqs. (2-17) and (2-18), and we can calculate θ and ϕ from them. We then find u and v using Eqs. (2-7) and (2-8).

2-3.2.1.5. Finding a Tangent to an Asp Edge

Given an asp edge in the form of Eqs. (2-7), (2-8), (2-21), and (2-22), we can find the slope at a point (θ,ϕ) on the edge by finding $d\phi/d\theta$. First we must find $ds/d\theta$ from Eq. (2-21). If we let $a = e_4 \tan\theta - e_1$, $b = e_5 \tan\theta - e_2$, and $c = e_6 \tan\theta - e_3$, this yields

$$\frac{ds}{d\theta} = \frac{(e_4\, b - e_5\, a)\sqrt{b^2 - 4\, a\, c} \pm a\,(e_5\, b - 2\, e_6\, a - 2\, e_4\, c) \pm (-e_4)(b^2 - 4\, a\, c)}{4\, a^2 \sqrt{b^2 - 4\, a\, c}\, \cos^2\theta} \tag{2-29}$$

Then we can find $d\phi/d\theta$ by taking the derivative of the right-hand side of Eq. (2-22), yielding

$$m = \frac{d\phi}{d\theta} = \frac{[v_y' (2 e_1 s + e_2) - v_x' (2 e_4 s + e_5)] \cos\theta \frac{ds}{d\theta} + v_x' v_y' \sin\theta}{v_x'^2 + v_y'^2 \cos^2\theta} \qquad (2\text{-}30)$$

Given some point (θ,ϕ) on a curve, we can plug (θ,ϕ) into Eqs. (2-29) and (2-30) to get the slope of the curve. A tangent vector is then <1,m>.

An alternative, approximate method of finding a tangent to a curve at a point is computationally more efficient. If the point in question is at the value s_0 of the parameter s along the curve, the method is to find the point on the curve at the value $s_0 + \varepsilon$, where ε is small. Then the difference between the points is a vector in the direction of the tangent.

2-3.2.2. Finding the Intersection of Asp Cells

In order to construct the asp for a face as occluded by other faces in front of it, we subtract the asp for the other faces from the asp for the given face. This subtraction can be accomplished as a complement and an intersection operation on the asp cells. In order to find the intersection of two asp cells we find the intersection of every pair of facets of the asps, and each time we find an intersection, we "glue together" the two asp facets at the intersection ridges. We then "cut away" the outer faces incident upon the ridges at which more than two facets meet. This process is illustrated for polygons in R^2 in Figs. 2-13 to 2-15. In Figure 2-13 we show overlapping polygons. The intersection points have been added in Figure 2-14, and the outer faces "cut away" in Figure 2-15.

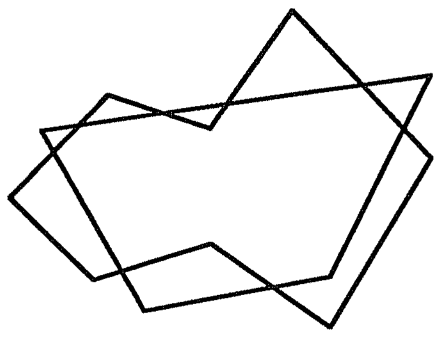


Figure 2-13. Overlapping polygons.

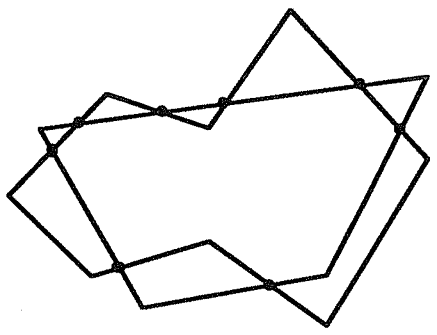


Figure 2-14. Intersection points added to overlapping polygons.

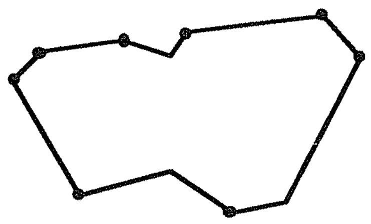


Figure 2-15. The intersection results when the outer faces are cut away.

In the case of polygons in the plane, finding the intersection point of a pair of edges involves finding the intersection point of the lines containing the edges and determining whether that point lies in both segments. If so, we then split the segments at the point and "cut away" the new half-segments that are in one of the polygons but not the other.

We describe the process for higher dimension by presenting an algorithm for the intersection of two k-polytopes in R^k. We state the algorithm recursively with four procedures:

Procedure 1: d-Polytope Intersection in R^d

Step 1: Find the intersection of all pairs of facets using Procedure 2.

Step 2: Cut away outer facets at ridges with four incident facets rather than the normal two.

Procedure 2: d–1-Polytope Intersection in R^d

Step 1: Find the intersection of the hyperplanes (i.e. d–1 spaces) containing the polytopes.

Step 2: If the hyperplanes are the same then solve the intersection problem in the hyperplane using Procedure 1 recursively.

Step 3: If the intersection is empty, return the empty set.

Step 4: If the intersection is a d–2-plane, find the intersection of the d–1-polytopes with the d–2-plane using Procedure 3 and solve the d–2-polytope in R^{d-2} intersection problem using Procedure 1.

Step 5: Split the two d–1-polytopes at their intersection hyperplane and insert their intersections into the data structures for the d–1-polytopes.

Procedure 3: d-Polytope, Hyperplane Intersection in R^d

Step 1: Find the intersection of each facet with the hyperplane using Procedure 4.

Step 2: The intersection will consist of d–2-polytopes in the hyperplane. Join them at common boundaries to form d–1-polytopes in the hyperplane.

Step 3: Return the list of d–1-polytopes.

Procedure 4: d–1-Polytope, Hyperplane Intersection in R^d

Step 1: Find the intersection of the hyperplane **h** containing the d–1-polytope **p** and the given hyperplane.

Step 2: If the hyperplanes are the same, return the d–1-polytope.

Step 3: If the intersection is empty, return the empty set.

Step 4: If the intersection is a d–2-plane, return the intersection of **p** with the d–2-plane in **h** using Procedure 3.

2-3.2.3. Constructing the Asp

Constructing the asp for a polyhedron involves constructing the asp for each face as occluded by the faces in front of it and taking the union. The asp for a face **f** as occluded by the faces in front of it is the asp for **f** minus the asps for the faces or parts of faces in front of it. The faces in front of **f** are found by computing the intersection of the plane containing **f** with the planes containing the other faces. The (directed) line of intersection of two planes is parallel to the cross product of outward normals to the planes. Then if every vertex is above the line in the plane (i.e. to the left in the direction of the intersection line), the face is in front of **f**. If every vertex is below the line in the plane, the face is behind **f**. If some vertices are above the line and some below, the intersection points of edges with the line are found and the polygon is cut at the intersection points. One or several polygons above **f** may result.

Once we have found the faces above **f**, we find the asps for the complements of those faces. We then take the intersection of **f** with each of those faces. The complements of the faces may be represented by setting a flag, so that the "cutting away the outer faces" step of the intersection algorithm cuts away the proper faces.

2-3.2.4. Analysis

Finding the intersection of two cells of aspect space involves finding the intersection of every pair of facets, finding the intersection of facets involves finding the intersection of every pair of ridges, and so on. Since the dimension is fixed, finding the intersection requires O(**nm**) time for two cells in which the total number of incidence relations of the form [facet, ridge, edge, vertex] is **n** and **m** respectively. That is, the algorithm requires O(**nm**) time where **n** and **m** are the sizes of the cells.

Constructing the asp for a polyhedron involves constructing the asp for each face; and constructing the asp for a face **f** of a polyhedron involves finding the intersection of a face with some or all of the other faces. If face **f** has size **m**, the number of visual events involving **f** and hence the size of the asp for **f** is O($\mathbf{mn}^3$) where **n** is the size of the original polyhedron. Therefore each intersection in the construction of the asp for the face takes time O($\mathbf{mn}^4$). The sum of the sizes of the faces of the polyhedron is O(**n**), so the construction time for the asp for a polyhedron is O($\mathbf{n}^5$).

2-3.3. Constructing an Image from the Asp for a Polyhedron

Given a viewpoint (θ,ϕ), the (θ,ϕ)–cross-section of the asp is the view of the polyhedron from that viewpoint. Thus, drawing the (θ,ϕ)–cross-section of the asp is equivalent to constructing an image of the polyhedron from that viewpoint with hidden lines removed. In this section we show how to find that cross-section of the asp for a polyhedron or polyhedral scene.

Each cell of the asp corresponds to a region of the image, which may be a face or part of a face of a polyhedron. The appearance of a part of the face of the polyhedron from viewpoint (θ,ϕ) is the cross-section of the cell from (θ,ϕ). The cross-section of the cell for a fixed viewpoint is a polygon since regions in an image corresponding to parts of faces of a polyhedron are always polygons. Each facet bounding the cell corresponds to an edge of the image polygon from some viewpoint, but not all of the facets correspond to visible edges of the polygon from (θ,ϕ). Thus, for the viewpoint

(θ,ϕ), we must find the facets that are visible, find the edges in the image that they correspond to, and draw the edges.

A facet is bounded by a number of ridges, corresponding to vertices in the image. Since the cross-section of a facet at a particular viewpoint is an edge in the image and all edges in the image have two visible endpoints, if the facet is visible then two of the ridges are defined at the viewpoint (that is, the projection of the viewpoint onto the surface containing the ridge is in the ridge). If the facet is not visible, none of the ridges will be visible. Thus, to determine whether a facet is visible from (θ,ϕ), it is sufficient to check all of the bounding ridges. If two of them are visible, then the points of intersection of those ridges with the (θ,ϕ)-plane are the endpoints in the image of an edge bounding a polygon.

It remains to determine whether a ridge is visible at a given viewpoint. Equivalently, we can project the bounding asp edges and vertices into viewpoint space and determine whether the viewpoint is in the region of viewpoint space that they bound. We present in detail below the algorithm for determining whether the point is in the region and therefore whether the object vertex is visible.

2-3.3.1. Determining Whether an Asp Ridge is Visible

An asp ridge is bounded by a cycle of asp edges and vertices; the edges are of the form of Eqs. (2-7), (2-8), and (2-22) above. Notice that Eq. (2-22) does not have any terms involving image plane coordinates. Thus we can ignore Eqs. (2-7) and (2-8) for the edges and consider Eq. (2-21) by itself as the projection of an edge onto viewpoint space. Determining whether a ridge is visible then amounts to determining whether a point is in a region bounded by edges of the form given by Eq. (2-21). The boundaries of the region are not arcs of great circles, so the region is not spherical polygonal.

A common algorithm for determining whether a point is in a polygon is to draw a ray from the point infinitely far in some direction and count the number of times it intersects the boundary of the polygon. If it intersects an odd number of times, the point is in the polygon. This algorithm must be modified for the case of a spherical polygon because a ray on a sphere does not go infinitely far in any direction: it goes around the sphere and back to the point. However, we do know that if a ridge is visible from some viewpoint, then a great circle from that viewpoint around viewpoint space and back to the same viewpoint will first cross the boundary *going out* of the spherical polygon if it crosses the boundary at all. Conversely, if the viewpoint is not in the spherical polygon, then the great circle will first intersect the boundary *going in* to the spherical polygon if it crosses the boundary at all. This suggests an algorithm for determining whether a point is in a spherical polygon: pick a great circle containing the viewpoint that intersects the boundary of the spherical polygon, find the intersection with the boundary that is closest to the viewpoint, and determine whether that intersection goes into or out of the polygon.

A great circle is the intersection of a sphere with a plane through the center of the sphere, so we find a great circle that intersects the boundary by taking the intersection of the sphere with the plane defined by the center of the sphere and two other points: the viewpoint in question and a point on one of the edges of the spherical polygon. This plane intersects viewpoint space in the required great circle. Representing viewpoint space as a unit sphere centered on the origin, the points defining the plane are (0,0,0), (θ_1,ϕ_1) (the test viewpoint), and (θ_2,ϕ_2) given by Eqs. (2-20) and (2-22) for one of the edges of the polygon, with some s chosen so that $0 < s < 1$. We transform the coordinates of the points to Cartesian coordinates, yielding $(\sin\theta\cos\phi, -\sin\phi, \cos\theta\cos\phi)$. A unit normal to the plane is given by the cross product of vectors to the two points on the sphere, i.e.

$$\mathbf{n}_1 = (\sin\theta_1\cos\phi_1, -\sin\phi_1, \cos\theta_1\cos\phi_1) \times (\sin\theta_2\cos\phi_2, -\sin\phi_2, \cos\theta_2\cos\phi_2) \tag{2-31}$$

We would like to find an equation for the great circle of viewpoints for θ in terms of ϕ. To do this we note that the dot product of the normal $\mathbf{n} = (n_x,n_y,n_z)$ and a viewpoint (θ,ϕ) on the great circle is zero. This yields

$$\tan\phi = \frac{n_x\sin\theta + n_z\cos\theta}{n_y} \tag{2-32}$$

In order to find the viewpoint of intersection of this great circle with an asp edge, we solve (2-32) and (2-22) for θ. This yields

$$\tan\theta = -\frac{v_y'\,n_y + v_x'\,n_z}{v_x'\,n_x} \tag{2-33}$$

We must then determine whether the points of intersection (θ,ϕ) thus calculated actually lie on the edge of the asp. We do this by calculating s from Eq. (2-21) and determining whether $0 \le s \le 1$. If so, (θ,ϕ) is an intersection point of the great circle and the edge.

As we find all of the intersection points of edges with the great circle, we can calculate their distance from the test viewpoint and keep track of the closest intersection point. We can use the Cartesian distance or the angle between the two points on the unit sphere. Once we find the closest intersection point, in order to determine whether the viewpoint is in the spherical polygon it remains to determine the sense of the intersection. That is, we must determine whether the great circle is going into or out of the projection of the ridge in viewpoint space. In order to determine that, we take the dot product of a tangent to the curve in the clockwise direction and the normal to the plane of the great circle. If the dot product is greater than zero, the great circle is going *into* the region of viewpoint space, so the point is not in the region. If the dot product is less than zero, the great circle is going *out* of the region, and the ridge is visible from that viewpoint.

To this point we have determined whether the test viewpoint is inside the region of viewpoint space. For each ridge of the asp, we make that test. Then for each facet, if it has two visible bounding ridges, we draw an edge in the image between the image points corresponding to those two ridges at the given viewpoint. The result of this process is the display of all of the visible edges of the polyhedron.

2-4. The Asp under Perspective Projection

Since in the perspective case we take viewpoint space to be R^3 rather than the sphere, aspect space is 5-D: $R^3 \times$ the image plane. To represent viewpoint space we use spherical coordinates (θ,ϕ,r), similar to the orthographic case but with the addition of distance r from the origin. Thus a point of aspect space is represented (θ,ϕ,r,u,v), where (θ,ϕ,r) is the viewpoint and (u,v) is the point on the image plane. The viewing direction is the direction from (θ,ϕ,r) to the origin. A point (θ,ϕ,r,u,v) of aspect space is in the asp for a polyhedron when (u,v) is in the image of the polyhedron from the viewpoint (θ,ϕ,r). In some cases we will also use Cartesian coordinates for viewpoints, $(v_x,v_y,v_z) = \mathbf{V}$.

Since a point $\mathbf{p} = (p_x,p_y,p_z)$ by itself in object space is visible from all viewpoints, it has as its asp one point in the image plane for each viewpoint $\mathbf{V}$. That is, the asp for a point is a 3-surface in aspect space. We can determine the point in the image plane in the same manner that we found the point in the orthographic case, except that we now use perspective rather than orthographic projection. In that case, after a rotation to make the viewpoint lie on the z-axis, the point $\mathbf{p}$ was given by Eq. (2-1). Under perspective projection, a point (p_x,p_y,p_z) projects to

$$\left[\frac{f}{d}p_x , \frac{f}{d}p_y\right]$$

where f is the focal length and d is the distance along the z-axis from the viewpoint to the object point, i.e. the difference in z-coordinates. Thus, the asp for the point (x_0,y_0,z_0) from the viewpoint (θ,ϕ,r) is given by

$$u = \frac{f\,(x_0 \cos\theta - z_0 \sin\theta)}{r - (x_0 \sin\theta \cos\phi - y_0 \sin\phi + z_0 \cos\theta \cos\phi)} \tag{2-34}$$

$$v = \frac{f\,(x_0 \sin\theta \sin\phi + y_0 \cos\phi + z_0 \cos\theta \sin\phi)}{r - (x_0 \sin\theta \cos\phi - y_0 \sin\phi + z_0 \cos\theta \cos\phi)} \tag{2-35}$$

The asp for a line segment is a 4-surface in aspect space. The equations for the asp can be obtained by substituting parametric equations for a line into Eqs. (2-34) and (2-35) for x_0, y_0, and z_0. The asp for a polygon is a cell of aspect space bounded by the 4-surfaces corresponding to the edges bounding the cell.

The asp for a polyhedron under the perspective model is much like the asp under the orthographic model. The difference is that the asp is a 5-manifold in a 5-D aspect space rather than a 4-manifold in 4-space. It can be constructed in the same way as in the orthographic case, if the algorithms for intersection and union are generalized to 5-D.

The asp faces corresponding to visual events are of one dimension higher in the perspective case. In Table 2-3 we show visual events and corresponding asp features for the perspective case. Note that there are no true 0-D visual events. A 0-D visual event would be a visual event visible at a particular point in the image, from only one viewpoint. However, for any visual event there is some line in viewpoint space along which the same visual event is visible, at points a little closer or further from the object. The only way that a 1-D visual event ends is when the line of sight passes through a face of a polyhedron in object space.

Visual Event	Asp feature
point	ridge (3-face)
edge	facet (4-face)
polygon	cell (5-face)
apparent int. point of two object edges	ridge
apparent intersection of edge and vertex	2-face
apparent intersection of three edges	2-face
apparent intersection of two vertices	edge (1-face)
apparent intersection of four edges	edge (1-face)

Table 2-3. Visual events and corresponding asp features under perspective projection.

The surfaces corresponding to these events are closely related to those in the orthographic case. There are two differences, resulting from the use of perspective rather than orthographic projection and from taking viewpoint space to be R^3 rather than the unit sphere. The viewing directions from which an event occurs in the perspective case are the same as those from which the event occurs in the orthographic case. However, all of the viewpoints along a line parallel to the viewing direction and intersecting an object point at which the event occurs must be taken as the set of viewpoints at which the event occurs. Thus the surface is of one dimension higher in the perspective case.

Under the perspective model asp faces correspond to the same visual events as they do under the orthographic model, except that the surfaces are in R^3 rather than on the sphere. Some visual events that were not visible in the orthographic case because they were behind other faces may be visible in the perspective case from some viewpoints in R^3. However, the asymptotic worst-case size of the asp and time to construct the asp is the same as the orthographic case since the same events are visible in the worst case.

2-5. Discussion

The aspect representation is a new, continuous, viewer-centered representation for polyhedral objects. It is viewer-centered in the sense that it represents the appearance of the object rather than the volume of space that it fills, and it is continuous in the sense that it represents appearance as a function of viewpoint rather than from a discrete set of viewpoints. The asp is the first example of a representation of appearance as a function of viewpoint.

The asp has something in common with multiview object representations, since any appropriate cross-section of the asp is a view of the object. However, it goes beyond multiview representations by representing appearance from all viewpoints continuously. Another difference is that multiview representations do not store any information about how different views relate to one another. The aspect representation makes clear exactly how "different views" of an object from different viewpoints (i.e. cross-sections of the asp) relate to one another.

An advantage of the aspect representation over object-centered representations is that by directly representing appearance, the asp makes it easier to answer questions about appearance such as at which viewpoints a particular visual event occurs or at which viewpoints a certain feature is visible. Using object-centered representations for such problems often requires testing all visual events to find the one in question. However, if they are pre-computed as in the asp, more efficient query-answering data structures and algorithms can be used.

The aspect representation is large since it represents all of the visual events for an object. Therefore it is generally useful only for problems that make repeated use of visibility information. Also, since all of the visibility information is made explicit, the asp is a natural choice for parallel algorithms, since all features of appearance can be dealt with in parallel.

The asp is smaller than multiview representations that represent all of the topologically-distinct views of an object. This may seem counter-intuitive since every view of the object is a cross-section of the asp. However, in the asp visual events are represented individually, rather than as a part of an image. Thus, rather than storing several distinct images, the asp in effect stores only the changes between them. A related drawback is that the asp does not contain information that allows quickly finding the nearest event to any viewpoint with a single processor.

Chapter 3. Constructing the Aspect Graph for Polyhedra

3-1. Introduction

Humans can recognize unexpected objects in around 100 neuron-firing times. That is a remarkable feat, considering the number of objects a typical adult is able to recognize and the amount of information contained in a single image. Computer vision will not duplicate that feat in the near future, but advances are being made. Currently the most widely-accepted paradigm of 3-D object recognition in computer vision is the "object-centered" approach outlined by Marr [1982], in which visible surfaces and boundaries are extracted from the image and an object-centered model of the object in the image is constructed. This object-centered model constructed from the image is matched with object-centered object models in the knowledge base for recognition.

It is difficult to imagine how a computer using such an approach could possibly recognize objects as quickly as a human, since many operations in the object-centered approach are inherently sequential. A different approach to object recognition that might possibly yield such fast results has been gaining in popularity recently. It is the "characteristic view" approach, in which several "typical" views of an object are stored, and all are compared to an image for recognition [Rosenfeld, 1987; Chakravarty and Freeman, 1982; Castore and Crawford, 1984; Ikeuchi, 1986; Korn and Dyer, 1987; Schneier *et al.*, 1986, Stewman & Bowyer, 1987]. The idea of the characteristic view approach is that if a view similar to the image is stored, the object can be recognized quickly, perhaps through comparisons of simple property measures such as relative position of parts, curvature, slope, number of extrema of zero-crossings of curvature, and so on [Rosenfeld, 1987]. The approach works particularly well in parallel, since the image can be compared with many characteristic views in parallel.

The views are often taken to be topologically distinct in the interest of minimizing the number of views necessary, with the idea that a topological match of the image and the view should be enough for recognition. In order to calculate these characteristic views one must calculate ranges of viewpoints with particular appearance characteristics, for example, maximal regions of viewpoints in which the object has the same "topological appearance." In this thesis we present the first general algorithm for constructing such views.

In order to compute these views, we catalog the ways that the *aspect* or topological appearance of a polyhedron can change with changing viewpoint. These changes we call *events*. The events give rise to boundaries in viewpoint space separating regions of viewpoints from which the aspect is different. Finding all of these boundaries enables us to construct a partition of viewpoint space into regions of constant aspect, which we call the *viewpoint space partition* (*VSP*).

We use the VSP to construct the aspect graph, a graph with a vertex for every aspect or topologically-distinct view of an object and with edges connecting adjacent aspects. We show how to do these things under two viewing models. The first uses orthographic projection and restricts viewpoints to a 2-D spherical space of directions, and the second uses perspective projection and a 3-D viewpoint space. We give tight bounds on the maximum sizes of the VSP and the aspect graph in all cases, thus determining the maximum number of topologically-distinct views of an object. In Table 3-1 we present a summary of the size of the VSP and aspect graph in various cases, and in Table 3-2 we present a summary of the time required to construct them in each case.

	convex polyhedra	non-convex polyhedra
orthographic	$O(\mathbf{n}^2)$	$O(\mathbf{n}^6)$
perspective	$O(\mathbf{n}^3)$	$O(\mathbf{n}^9)$

Table 3-1. The maximum size of the VSP and aspect graph.

	convex polyhedra	non-convex polyhedra
orthographic	$O(\mathbf{n}^2)$	$O(\mathbf{n}^6 \log \mathbf{n})$
perspective	$O(\mathbf{n}^3)$	$O(\mathbf{n}^9 \log \mathbf{n})$

Table 3-2. Construction time for the VSP and aspect graph.

In Section 3-2 we describe the viewing models and discuss the notions of visibility and occlusion. In Section 3-3 we define the VSP and the aspect graph and discuss some of their properties. In Section 3-4 we give upper and lower bounds on the maximum size of the VSP and the aspect graph for convex polyhedra and present an algorithm for their construction. We do this first under the orthographic viewing model and then the perspective model. In Section 3-5 we consider the same problems for the case of non-convex polyhedra. We characterize the types of boundaries of changing aspect that occur for non-convex polyhedra and present upper and lower bounds on the maximum size of the VSP and the aspect graph. We then present an algorithm for constructing the VSP and the aspect graph. We do this first under the orthographic model and then the perspective model. In Section 3-6 we make concluding remarks.

3-2. Visibility and Occlusion

Visual perception involves acquiring information about the surrounding world from an image, i.e. a representation of the array of ambient light available at a viewpoint. A surface in the world is said to be visible in an image whenever light arrives at the viewpoint from the surface, so that it is represented by a region of light intensity values in the image. However, in some cases it is desirable to define a more restrictive model of visibility, in order to focus attention on a particular aspect of the problem of visual perception.

In this chapter we are interested in the change in the topological structure of images of polyhedra, and we use two models of visibility that enable us to study the problem in two slightly different ways. In both models the world contains solid, opaque polyhedra and the edges of the polyhedra are visible in the image. Thus, the image plane is an unbounded plane containing line segments corresponding to the projections of the visible edges of the polyhedra in the world. We assume that the resolution of the image is infinite; nothing is too small to resolve. Note that under these models the image of a polyhedron is more than a silhouette; it is a line drawing with hidden lines removed. Equivalently, the image can be thought of as a set of polygonal regions corresponding to visible faces or parts of faces of the polyhedron.

Since the image plane only captures incident light from some directions (half of them) and some part of the image plane may be preferred for objects of interest (e.g. the fovea), we also speak of a *viewing direction*. The viewing direction or line of sight is the direction in which the camera is pointed, so that objects in the viewing direction appear at the preferred point of the image. We also assume that the camera has a fixed orientation around the viewing direction with respect to the image, so that objects appear upright.

For some problems it is sufficient to assume that the viewpoint is a large distance away from the object of interest and that the whole object is visible in the image. For example, in object recognition it may be reasonable to assume that the whole object to be recognized appears in the image. In that case, the viewpoint must not be inside the object to be recognized or so close to the object that only one face is visible, for example. For other problems the space of possible viewpoints must range over the whole world. For example, in robot motion planning the visible surfaces represent the boundaries of free space in any direction. In many motion planning problems, the object (or viewpoint) might be anywhere in the world.

Thus we use two viewing models. The first model is designed to handle the first case, in which the whole object is in front of the viewer in any image. In order to model this case, we effectively restrict viewpoints to an infinite sphere containing the object and use orthographic projection in forming the image. Thus the whole object is in front of the viewer from any viewpoint. We call this the *orthographic model.* It is a useful model because it represents a restriction on visibility that makes some representations smaller. In the second model we allow the viewpoint to be any point in the world. In this case it is natural to use perspective projection in forming the image from a viewpoint. This model, called the *perspective model,* is intended to resemble natural vision, in which the viewpoint is unrestricted, and the viewer can even be inside objects, such as houses.

The directions of incident ambient light represented in the image plane are those directions more than 90° away from the viewing direction (since the viewing direction is *away* from the viewer but the visible light is *toward* the viewer). In problems such as motion planning, visibility information in all directions is of interest, since the visible surfaces represent the boundaries of free space in any direction. Visibility in all directions from the viewer can be represented by two image planes from opposite viewing directions along with points at infinity in one of the planes or by an image *sphere* representing all viewing directions. For the problem of looking at an object and recognizing it or perceiving its shape, however, a section of a single image plane is usually sufficient.

In the perspective model, viewpoint space is R^3 and the viewing direction is the direction from the viewpoint to the origin. Thus, the image formation process can be modelled mathematically as a perspective projection of the visible edges of the object onto the image plane, in such a way that the origin of the world projects onto the origin of the image plane. The object of interest will project onto a region of pixels near the origin of the image.

In the orthographic model, we need only represent the *direction* of the viewpoint, rather than its location in the world. In this model viewpoint space is the 2-D space of directions from the origin, the same space as defined in Section 2-2 (see Figure 3-1).

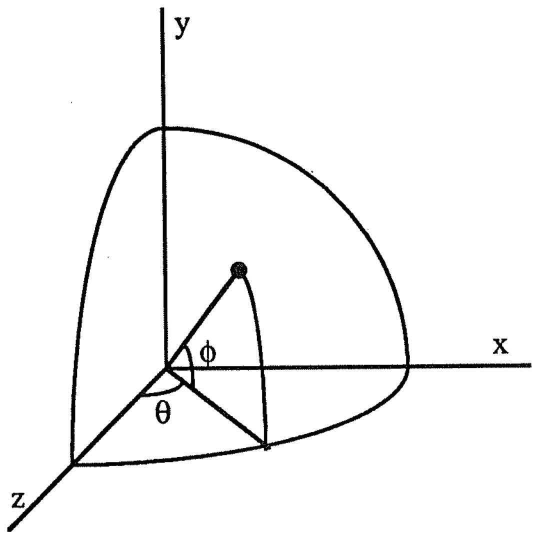


Figure 3-1. The point (θ,ϕ) on the unit sphere.

Occlusion is the general term for the hiding of some feature in the world from some viewpoint. That is, when the view of something is blocked from a particular viewpoint, we say that it is occluded. In this chapter we are concerned with the changes in occlusion that bring about a topological change in the image. A topological change in the image results when the structure of line segments and regions changes. This occurs when a region appears or disappears, when regions split or merge, or when the number of edges in a region increases or decreases. For example, the merging of two regions of the image results in a topologically-distinct image (see Figure 3-2). All of the kinds of topological changes are listed in Section 3-5.

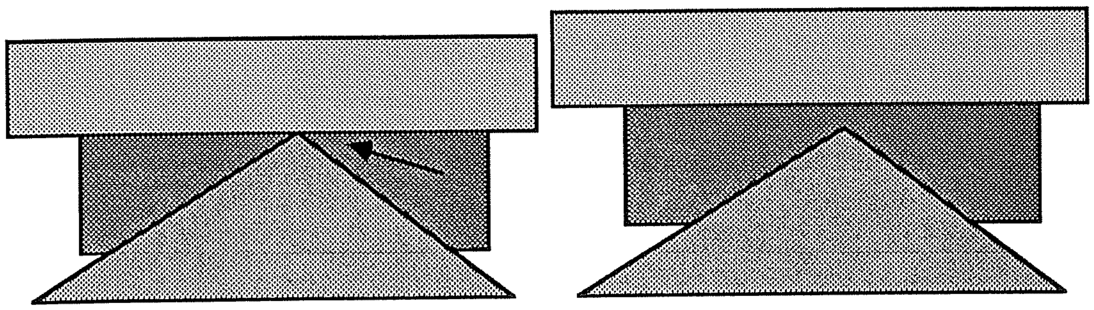


Figure 3-2. The merging of two regions in the image is a topological change.

The changes in occlusion that result in a topological change in the image are called *events*. These are the events that together form the VSP. In this chapter, we study the ways in which the structure of regions and edges in the image of polyhedra changes with changing viewpoint. We characterize the way that the structure of the image changes by studying the viewpoints at which such changes occur.

Many slightly different definitions of topologically-distinct images are possible. Images might be said to be topologically distinct when the set of visible faces changes along every path of viewpoints between them or when the set of regions in the image changes along every path of viewpoints between them. Both of these alternate definitions are useful for different problems. The algorithm presented below for constructing the aspect graph will work with any of these definitions; the only way that the algorithm would have to be modified is in the selection of the events listed in Section 3-5 that are considered to be topological changes.

3-3. The Viewpoint Space Partition and the Aspect Graph

From any viewpoint the image of a polyhedron has a particular structure of regions and edges, and varying the viewpoint slightly does not generally change the structure or topology of the image. Thus, there are "stable views" or *aspects* for a polyhedron—views of the polyhedron with the same topological image structure over a region of viewpoints. We call a maximal connected region of viewpoints from which the views of the object are topologically equivalent a *maximal viewing region of constant aspect* or *viewing region*. Some changes in viewpoint result in a change in the topology of the image, however. These changes pass from one viewing region to another.

We can define a viewing region more formally as a set of viewpoints that are equivalent under the following equivalence relation: for two viewpoints $\mathbf{v}_1$ and $\mathbf{v}_2$, we say that $\mathbf{v}_1 \approx \mathbf{v}_2$ whenever there is a path of viewpoints from $\mathbf{v}_1$ to $\mathbf{v}_2$ such that from every viewpoint along the path (including $\mathbf{v}_1$ and $\mathbf{v}_2$) the aspect is constant, i.e. the image has the same topological structure. $\approx$ is an equivalence relation since it is reflexive, symmetric, and transitive. Therefore it partitions viewpoint space into equivalence classes consisting of regions or volumes of constant aspect, i.e. viewing regions.

We call this partition of viewpoint space the *viewpoint space partition* or *VSP*. In the orthographic case it consists of a partition of the sphere, and in the perspective case it consists of a partition of R^3. The boundary points of the VSP are points from which an arbitrarily small change in viewpoint causes a change in the topology of the image. Figure 3-3 shows a tetrahedron and Figure 3-4 shows the structure of the viewpoint space partition generated by the tetrahedron under orthographic projection. The viewpoint space partition is flattened out onto a plane, and the shapes of the regions are not preserved, but the topological structure is preserved.

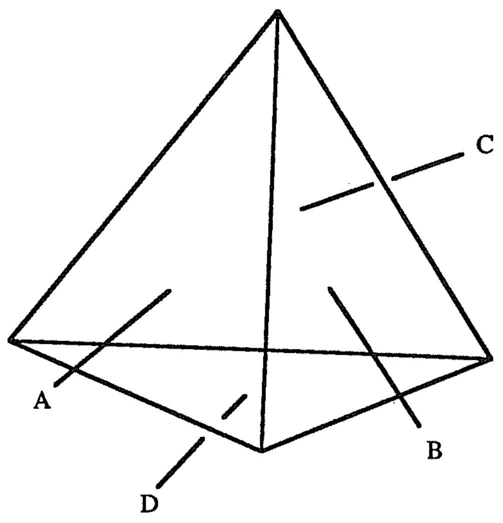


Figure 3-3. A tetrahedron.

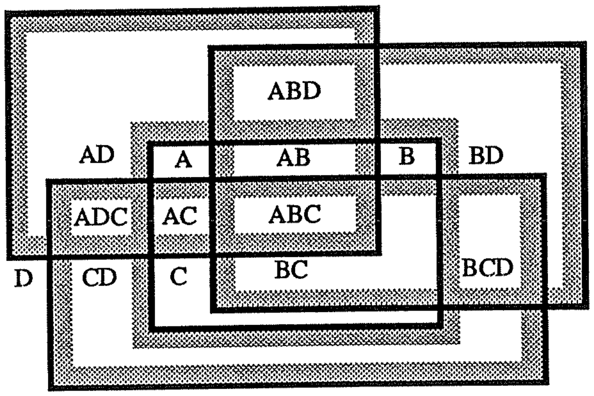


Figure 3-4. The viewpoint space partition for a tetrahedron.

The shaded side of a boundary represents the inside of a region. Each region is labelled by the faces visible from viewing directions in that region. Since a face is not visible edge-on, the regions are open and their complements are closed.

The definition of the VSP does not include labels for the regions. They are included in Figure 3-4 to aid understanding the figure. If one wishes to label the regions with the corresponding aspects, labeling the regions with the names of the visible faces is sufficient in the convex case, as we have done in Figure 3-4. It is not sufficient in the general case, however, since images with the same sets of visible faces can have different topological structure. In that case we can label a region by adding a pointer to an image of the object from a viewpoint in that region. Alternatively, we could label the boundaries with a description of the change that occurs in the image.

We define the dual of the region-boundary structure of the VSP to be the *aspect graph*. That is, for every region of the VSP there is a vertex in the aspect graph, and vertices of the aspect graph are connected by edges whenever the corresponding regions of the VSP are adjacent. Thus, the aspect graph has a vertex for every aspect, and vertices for adjacent aspects are connected. Figure 3-5 shows the aspect graph for the tetrahedron of Figure 3-3 overlaid on the VSP, and Figure 3-6 shows the aspect graph by itself. In this case the aspect graph is the same under the orthographic and perspective models.

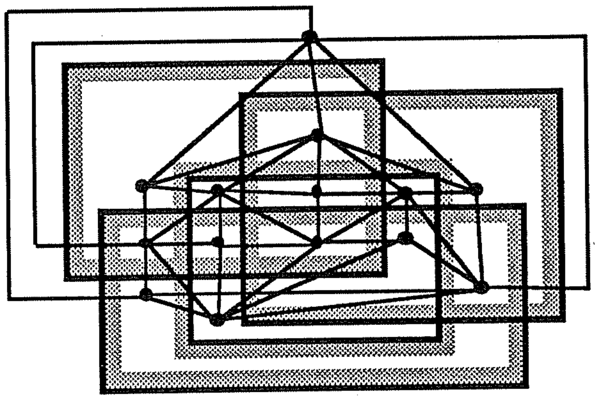


Figure 3-5. The aspect graph is the dual of the VSP.

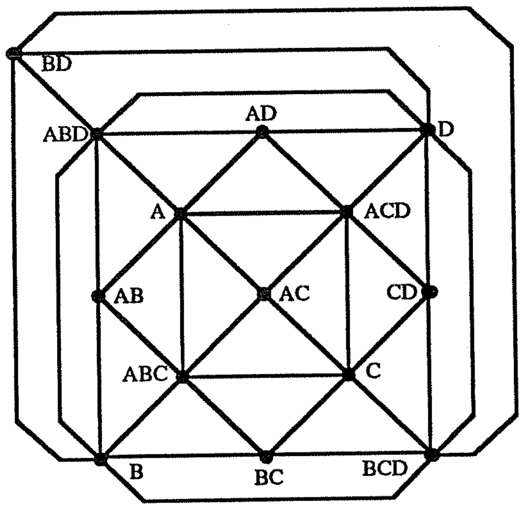


Figure 3-6. The aspect graph for a tetrahedron.

In Figure 3-6 the vertices of the aspect graph are labelled by the names of the faces visible from the corresponding aspect. Again, the labels are not included in the definition of the aspect graph. In order to associate the aspect with vertices of the aspect graph, it is sufficient in the convex case to label each vertex with the names of the faces visible from a viewpoint in the corresponding region. In the non-convex case this is not sufficient, and one can label the vertices with a corresponding image of the object or the edges with a description of the change that occurs.

Note that under our definition of the VSP and the aspect graph two different vertices of the aspect graph can correspond to aspects that are topologically equal.

This happens when the appearance of the object is the same from two different, unconnected, maximal regions of viewpoint space. Furthermore, the appearance from the two aspects can be the same in two different senses. In the more restrictive sense, the same set of faces may be visible and have the same topology from a different region of viewpoint space. In the less restrictive sense, a completely different set of faces may have the same topological appearance. These are also reasonable notions of the identity of aspects, but they differ from our notion.

Since the aspect graph has a vertex for every topologically-distinct view of a polyhedron, it is a tool for characterizing the distinct views of an object. It has been used in computer vision for that purpose. However, the VSP represents essentially the same information with the addition of the actual boundaries in viewpoint space of the viewing regions, and the space it requires is only a constant factor larger. Since the VSP represents viewing regions and their boundaries, it is a more appropriate tool for most applications that are concerned with the particular viewpoints associated with each aspect.

Notice that in Figure 3-6 vertices A and ABC are connected, whereas vertices AB and AC are not. This may seem inconsistent, since the corresponding regions of viewpoint space seem to have the same relationship to each other in both cases: paths of viewpoints connecting a viewpoint in one region with a viewpoint in the other all pass through a single point P (see Figure 3-7).

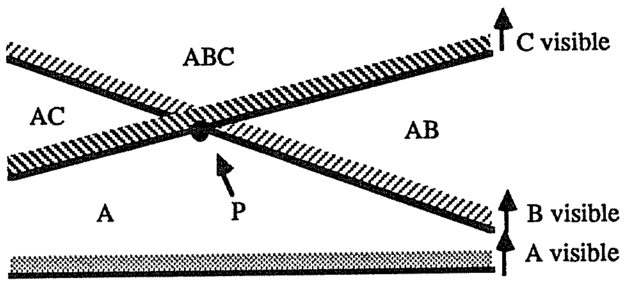


Figure 3-7. The boundary between four regions of viewpoint space.

The reason for the difference is that faces are not visible edge-on. The regions of viewpoint space in Figure 3-7 are shown with shading on open boundaries. Thus, the point P is in region A but not in regions AB, AC, or ABC. Therefore it is possible to find a path of viewpoints from a point in region A to a point in region ABC which doesn't go through AB and AC, but it is not possible to find such a path between AB and AC.

Koenderink and van Doorn [1976] first introduced the notion of stable views and visual events in a study of the singularities of the visual mapping of a smooth object and a moving viewer. In a later paper [1979], these stable views are termed "aspects" and the *visual potential graph* is defined, the equivalent of our aspect graph. Their definition of aspect is slightly different from ours, since they are interested in the singularities of the visual mapping for smooth objects. They state that an aspect contains boundaries (contours, occluding contours, cusp points, T-junctions, and maxima of distance increase on the occluding contours) and singular points and paths (specular points, separatrices, semicorner paths, lines of steepest increase through a spine point, and lines of steepest increase that touch an occluding contour).

In contrast, only edges of polyhedra are visible in our model, and aspect changes when the structure of the edges in the image changes. However, many of the elements of Koenderink and van Doorn's definition of aspect do not appear for polyhedra, and in fact they give an example of the visual potential graph (or aspect graph) for a tetrahedron. The visual potential graph they show for a tetrahedron is similar to (and the inspiration for) the aspect graph as used in this chapter.

3-4. Convex Polyhedra

In this section we consider the VSP and aspect graph for a single convex polyhedron. The restriction to convex polyhedra greatly simplifies the problem because a face is visible from all viewpoints "in front of" it; no face ever occludes another face. The only way that a face may become invisible with changing viewpoint is by "turning away from" the viewer or "going below the horizon." That is, when the viewpoint passes through the plane containing the face, the face becomes invisible (or visible) due to occlusion of the face by the rest of the object. Also, there cannot be two vertices in the aspect graph for the same set of faces since if the same set of faces is visible from two viewpoints, that set is also visible from all viewpoints along the shortest path (geodesic) in viewpoint space connecting them. Thus, in the convex case a change in aspect and a change in the visibility of some face of the polyhedron are equivalent events.

3-4.1. Orthographic Case

Under the orthographic viewing model, viewpoint space is a 2-D spherical space of directions. A face of a convex polyhedron is visible in any viewing direction "pointing toward" the front of that face. That is, if $\mathbf{d}$ is a viewing direction and $\mathbf{n}$ is the outward normal of the face, that face is visible whenever $\mathbf{n} \cdot \mathbf{d} < 0$, i.e. the angle between $\mathbf{n}$ and $\mathbf{d}$ is greater than 180°. Therefore the boundary between the regions of the sphere of viewing directions where the face is visible and where it is invisible is the great circle defined by $\{\mathbf{d} : \mathbf{n} \cdot \mathbf{d} = 0\}$. This is the intersection of the sphere with a plane containing the origin and having normal $\mathbf{n}$.

3-4.1.1. An Upper Bound

The great circle corresponding to a face is the boundary on the sphere of viewpoints between the region (hemisphere) where the face is visible and the region where it is invisible. For a polyhedron with **n** faces, the corresponding **n** great circles therefore partition viewpoint space into a subdivision in which the same set of faces is visible from all of the viewpoints in each region. That is, the **n** great circles partition viewpoint space into the VSP.

Assuming that no two faces have antiparallel normals, each pair of great circles intersects in exactly two points. In that case the total number of intersection points is at most $2\,\mathbf{n}\,(\mathbf{n}-1) = O(\mathbf{n}^2)$. (The number can be less than $2\,\mathbf{n}\,(\mathbf{n}-1)$ if several great circles intersect in the same point.) If two faces of the polyhedron are parallel, the corresponding great circles coincide, and the number of vertices in the VSP is reduced. Thus the VSP has $O(\mathbf{n}^2)$ vertices. Since a subdivision of a 2-D space is by definition planar and a planar graph has a linear number of edges and regions, the size of the VSP is therefore $O(\mathbf{n}^2)$. The aspect graph is the dual of the VSP, so it is also planar and also has size $O(\mathbf{n}^2)$.

3-4.1.2. A Lower Bound

The lower bound on the maximum size of the VSP and the aspect graph for a convex polyhedron of size **n** under orthographic projection is $\Omega(\mathbf{n}^2)$. We prove this by exhibiting a class of such polyhedra. Consider a band of **m** square faces arranged around a circle (see Figure 3-8). In Figure 3-9 we show two such bands arranged orthogonally around a sphere, with additional faces added to form a convex polyhedron.

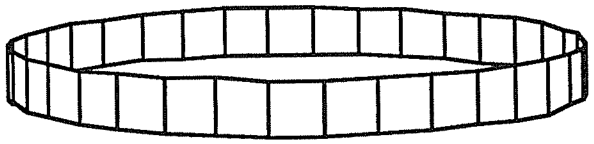


Figure 3-8. A band of m sides.

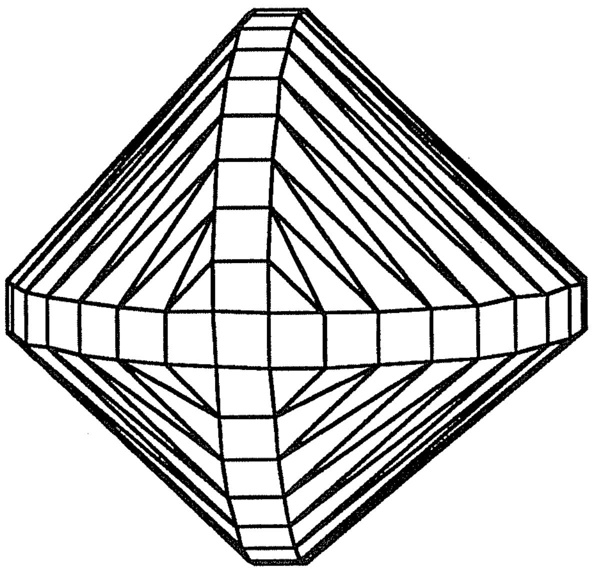


Figure 3-9. A convex polyhedron with two orthogonal bands.

Select one of the faces visible in Figure 3-9 from each band, excluding the face that is common to both bands. The great circles bounding the visibility regions on the sphere of the two selected faces have two intersection points, and these points are vertices of the VSP. Each pair of faces thus selected has a unique pair of intersection points, so there are $\Omega(\mathbf{m}^2)$ vertices in the VSP. The faces added to the bands to form a polyhedron have total size $O(\mathbf{m})$, so if the polyhedron has size $\mathbf{n}$, its VSP has size $\Omega(\mathbf{n}^2)$. Since the aspect graph is the dual of the VSP, it also has size $\Omega(\mathbf{n}^2)$.

In fact, a polyhedron would have to be highly degenerate to have a VSP of size less than $\Theta(\mathbf{n}^2)$. For example, any class of polyhedra that has bounded vertex degree and bounded face degree has a VSP of size $\Omega(\mathbf{n}^2)$. This is because the only way to reduce the size of the VSP is to have the intersection points of the great circles coincide, and if the number of edges around each face and the number of faces around each vertex is bounded, then the number of great circles that can intersect in a single point is bounded. An example of a highly degenerate class of polyhedra is the class of $\mathbf{n}$-sided approximations to a cylinder; for polyhedra in this class the VSP has size $O(\mathbf{n})$.

3-4.1.3. An Algorithm

The great circles defined by the face normals subdivide the sphere of viewing directions into the VSP. Therefore to give an algorithm for constructing the VSP it is only necessary to show how to merge these great circles into one subdivision data structure. This data structure represents regions, edges, and vertices of the subdivision and adjacencies of the various items. Edelsbrunner *et al.* [1986] present an algorithm for constructing an arrangement (subdivision) formed by $\mathbf{n}$ lines in the plane in $O(\mathbf{n}^2)$ time. The algorithm for constructing the arrangement can be used for constructing a subdivision formed by great circles on the sphere since they behave very much like lines on a plane. In fact, the great circles can be projected onto lines on two parallel planes and the arrangements constructed in the planes. This approach is similar to that of McKenna and Seidel [1985], in which they construct minimal or maximal shadows of a convex polytope generated by a point light source at infinity.

The algorithm for constructing the arrangement works by starting with an empty subdivision and adding one line at a time. Adding a line **l** to the subdivision involves finding an intersection point of **l** with a line already in the subdivision. Both lines are split at the intersection point: the nodes in the subdivision for the edges that intersect are duplicated, and one endpoint of each of the four edges is set to the intersection point. **l** crosses a region of the subdivision in both directions from that point. The edges of one of the regions are tested for intersection with **l**, and the intersection point thus found is added to the subdivision. The region just crossed by **l** is split into two regions, one on either side of the new edge on **l**. This process is continued in both directions from the first intersection point of **l** with an edge of the subdivision until all of **l** has been inserted into the subdivision.

This algorithm constructs the arrangement defined by **m** lines in a plane in $O(\mathbf{m}^2)$ time. Thus the VSP for a convex polyhedron can be constructed in $O(\mathbf{n}^2)$ time, which is worst-case optimal. In fact, the algorithm is optimal for all but highly degenerate polyhedra, since only in that case does the VSP have size smaller than $\Theta(\mathbf{n}^2)$.

The aspect graph is the dual of the region-edge structure of the VSP (see Figure 3-5). That is, the aspect graph has a vertex for every region of the VSP and edges connecting vertices corresponding to adjacent regions. Thus the aspect graph can be constructed by copying the region-edge subgraph of the VSP, changing the regions to vertices, and changing the edges separating regions to edges connecting adjacent vertices. This can be done in linear time in the size of the VSP, so constructing the aspect graph can be done in $O(\mathbf{n}^2)$ time.

3-4.2. Perspective Case

Under the perspective model the viewpoint is not restricted, so the viewpoint space is R^3. Also, objects are not restricted to being in front of the viewpoint; they can be in front of or behind the viewer. The viewpoint can even be inside a solid object, but in that case nothing is visible since only the front side of a face is visible. Thus the image changes as the viewpoint moves in any one of three independent directions rather than two. Therefore the VSP and aspect graph are much larger in this case. The reason for the larger size is the higher dimensionality of viewpoint space and not the change from orthographic to perspective projection.

The only kind of occlusion that occurs in this case is a face turning away from the viewer, as in the orthographic case. However, under the perspective model a face is visible from all points in front of the plane spanned by that face (rather than from all viewing directions pointing toward that face). Therefore viewpoint space is cut by **n** planes for a polyhedron with **n** faces, and the resulting subdivision is the VSP. The aspect graph is the dual of the cell-face structure of the VSP.

In fact, the aspect graph for a polyhedron under perspective projection contains as a subgraph the aspect graph for that polyhedron under orthographic projection. This is true because in the limit of increasingly large spheres, the planes spanned by the faces of the polyhedron cut the sphere in the same great circles as in the orthographic case.

3-4.2.1. An Upper Bound

The VSP is the subdivision of viewpoint space generated by the **n** planes corresponding to the **n** faces of the polyhedron. Since the intersection of three general planes is a point, there can be at most $O(\mathbf{n}^3)$ intersection points of **n** planes. In order to get the largest possible VSP, we can assume that every set of three planes intersects in a distinct point. These intersection points are vertices of the VSP, so the VSP has $O(\mathbf{n}^3)$ vertices.

At worst, each plane is intersected by all the others, resulting in $O(\mathbf{n})$ lines subdividing each plane. Thus each plane is cut into a subdivision of $O(\mathbf{n}^2)$ vertices, edges, and regions. Therefore among all **n** planes there are $O(\mathbf{n}^3)$ vertices, edges, and faces. A cell of the VSP is bounded by only four faces in the worst case, so the VSP has at most $O(\mathbf{n}^3)$ vertices, edges, faces, and cells. The aspect graph is the dual of the cell-face structure of the VSP, so its size is also $O(\mathbf{n}^3)$.

3-4.2.2. A Lower Bound

The lower bound example of the orthographic case (see Figure 3-9) is also a lower bound in the perspective case. Consider the faces of the two bands in any octant, i.e. one quarter of each band in the same octant (see Figure 3-10).

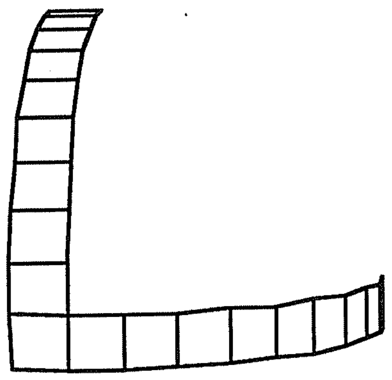


Figure 3-10. Two quarter-bands.

Any three distinct faces from these band-quarters (two from one band and one from the other) determine a distinct vertex in the VSP since they have a distinct intersection point. There are $\Omega(\mathbf{n})$ faces in each band quarter, so the VSP has $\Omega(\mathbf{n}^3)$ vertices. The VSP must also have $\Omega(\mathbf{n}^3)$ cells, so the aspect graph also has size $\Omega(\mathbf{n}^3)$. As in the orthographic case, a class of polyhedra would have to be highly degenerate to have size less than $\Theta(\mathbf{n}^3)$. For example, any class of convex polyhedra with bounded vertex degree and face degree has a VSP of size $\Omega(\mathbf{n}^3)$.

3-4.2.3. An Algorithm

The **n** planes corresponding to the **n** faces of a polyhedron partition space into the VSP. In order to construct the VSP, we must find all of the intersections of the planes and construct the data structure for the VSP by finding its cells, faces, edges, and vertices and linking incident faces. The algorithm of Edelsbrunner *et al.* [1986] for constructing arrangements of hyperplanes does the job in optimal, $O(\mathbf{n}^3)$ time. The algorithm builds the subdivision one plane at a time, as in the orthographic case. A plane **p** can be added to the subdivision by finding the line of intersection of **p** and each plane already in the subdivision. The line of intersection of **p** and another plane **q** already in the subdivision is then inserted into the 2-D subdivisions on **p** and **q** using the 2-D subdivision algorithm of the orthographic case. After the 2-D subdivisions have been constructed, the cells of the 3-D subdivision can be added by traversing the subdivision data structure.

The aspect graph is the dual of the cell-face graph of the VSP. It can be constructed from the data structure for the VSP by copying the VSP, changing cells to vertices and faces between cells to edges joining them, and deleting everything else. This algorithm requires linear time in the size of the VSP, so the aspect graph can be constructed in $O(\mathbf{n}^3)$ time.

3-5. Non-Convex Polyhedra

In the convex case, the only sort of event that can change the aspect is the horizon effect. In the non-convex case, faces can occlude other faces, causing several other types of events. Thus, in order to construct the VSP and the aspect graph in the non-convex case we must characterize all the ways in which aspect can change. Since there are more modes of changing aspect, the size of the VSP and the aspect graph for non-convex polyhedra can be much larger. In the following sections we list the kinds of events that can occur and the corresponding boundaries of regions of constant aspect that result in viewpoint space.

3-5.1. Orthographic Case

In the orthographic case, the VSP is a subdivision of the sphere of viewpoints, and the aspect graph is its dual. We first list the ways that the aspect can change in the non-convex case and the kinds of boundaries in viewpoint space that these changes generate. We then present upper and lower bounds on the maximum size of the VSP and aspect graph in this case and present an algorithm for their construction.

3-5.1.1. Occlusion under Orthographic Projection

In order to compute the regions of constant aspect in viewpoint space, we compute the events or boundaries of occlusion, that is, the boundaries of these regions. We define a boundary viewpoint to be a viewing direction from which there exists an arbitrarily small change in viewpoint that suffices to change the topology of the image. In this section we list several kinds of events in viewpoint space. At these events, regions of the image appear, disappear, split, merge, or gain or lose edges. We then show that these are the only kinds of events that occur for polyhedra. The types of events are horizon boundaries (which were discussed in the convex case), edge-vertex events, and edge-edge-edge events.

3-5.1.1.1. Edge-Vertex Events

Edge-vertex events occur at viewpoints from which a vertex and an edge of the object appear at the same point in an image. These are boundaries of occlusion since a small change in viewpoint can change the topological structure of the image. The image can change by the appearance or disappearance of a region in the image, by the splitting or merging of regions, or by a change in the number of edges of a region.

The appearance or disappearance of a region occurs when a small change in viewpoint makes a face visible or invisible at an image point. An example is shown in the fourth row of Figure 3-11, below. Regions can also split or merge because of a small change in viewpoint. An example of such a change is shown in the second row of Figure 3-11, below. Finally, regions may lose or gain edges, as shown in the first row of Figure 3-11. All of the distinct edge-vertex events in which the vertex is in front of the edge are shown in Figure 3-13.

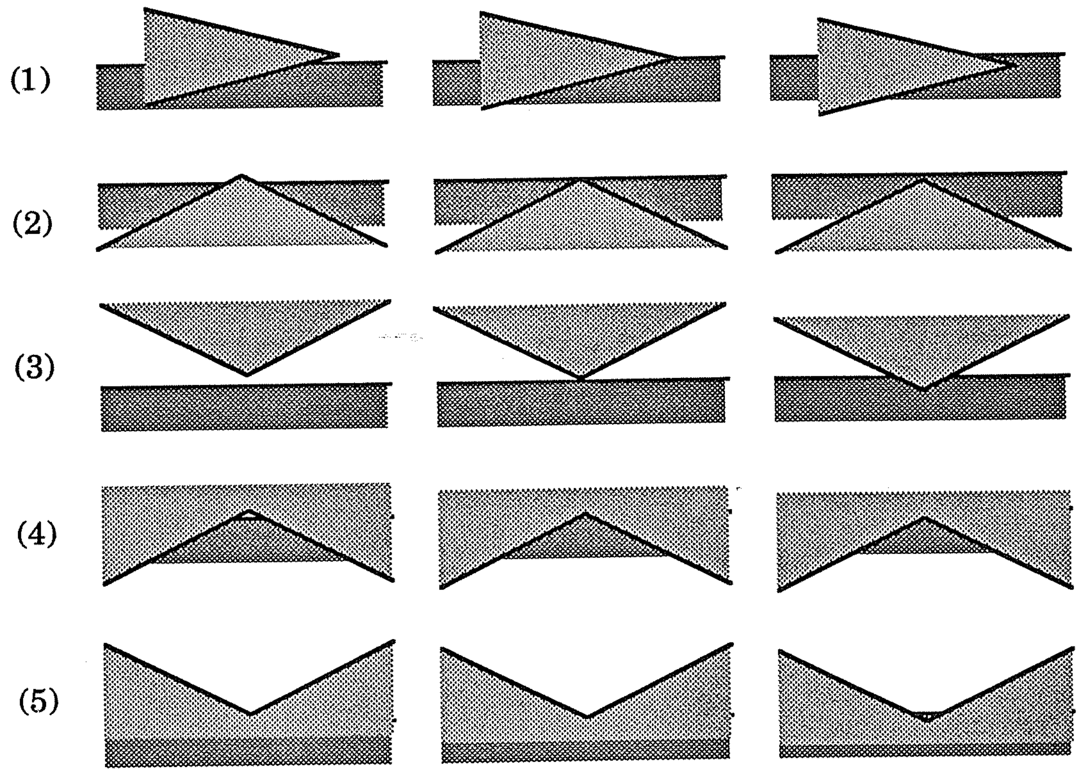


Figure 3-11. Edge-vertex events with the vertex in front of the edge. In each row, the center figure shows the event and the left and right figures show the aspect on either side.

The five rows in Figure 3-11 show five different sorts of events. In each row, the center figure shows the event, and the other figures show the aspect on either side of the event. For example, in the first row the number of edges in each region changes across the event. In the second row, the number of edges changes on the left and two regions merge on the right. Figure 3-12 shows the five kinds of edge-vertex events that occur when the vertex is behind the edge.

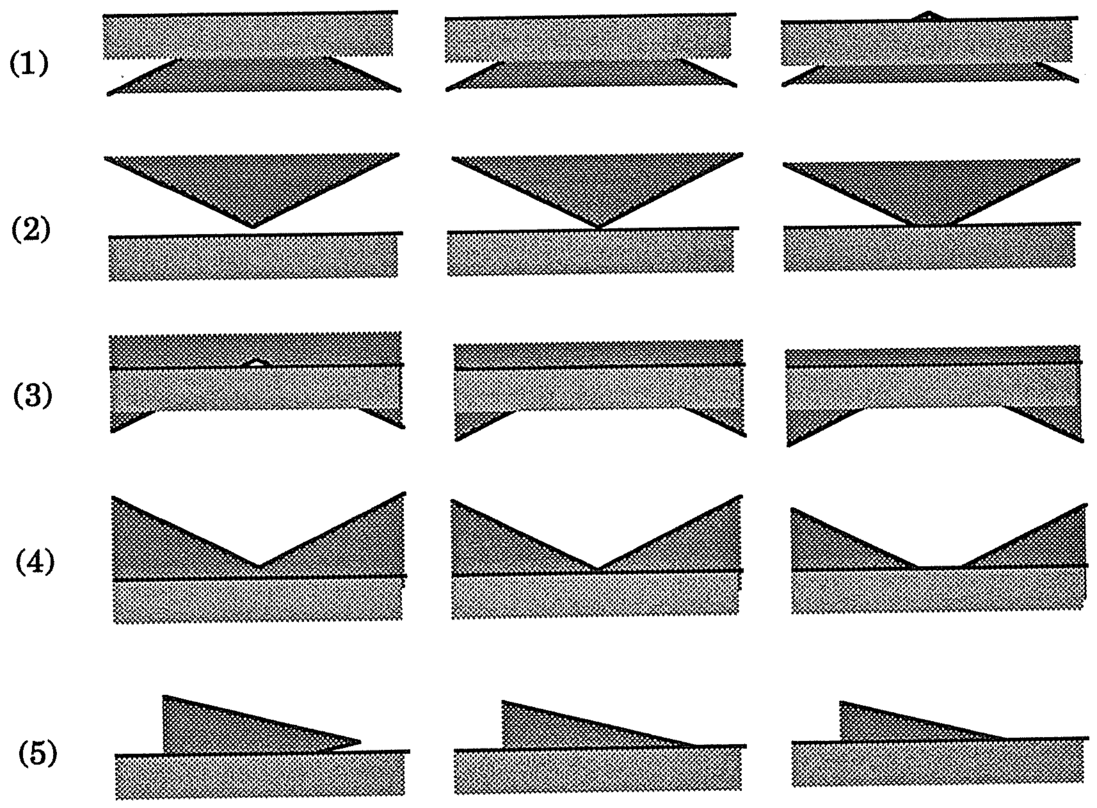


Figure 3-12. Edge-vertex events with the vertex behind the edge. In each row, the center figure shows the event and the left and right figures show the aspect on either side.

Boundaries in viewpoint space of edge-vertex occlusion occur at viewpoints such that a vertex of one face is directly in line with an edge of another face in the image. Therefore the only candidate viewpoints for such events are viewing directions parallel to the plane containing the point and the edge. That is, for a point $\mathbf{p}$ and an edge from $\mathbf{p}_1$ to $\mathbf{p}_2$, the candidate viewpoints are the points on the arc of the great circle with the normal $(\mathbf{p}_1 - \mathbf{p}) \times (\mathbf{p}_2 - \mathbf{p}_1)$. The boundaries of the arc are the points of intersection with the lines $(\mathbf{p}, \mathbf{p}_1)$ and $(\mathbf{p}, \mathbf{p}_2)$. Since there are $O(\mathbf{n})$ vertices and $O(\mathbf{n})$ edges in a polyhedron or polyhedral scene, there are $O(\mathbf{n}^2)$ edge-vertex pairs, so this type of occlusion is responsible for generating at most $O(\mathbf{n}^2)$ events (arcs of great circles) on the sphere of viewpoints.

3-5.1.1.2. Edge-Edge-Edge Events

Another type of event occurs at viewpoints where three unconnected object edges appear to intersect in a single image point. We call this edge-edge-edge occlusion. The three distinct kinds of edge-edge-edge events are shown in the three rows of Figure 3-13. In the first and second kinds, a region appears or disappears in the image, and in the third kind, regions lose or gain edges.

(1)

(2)

(3)

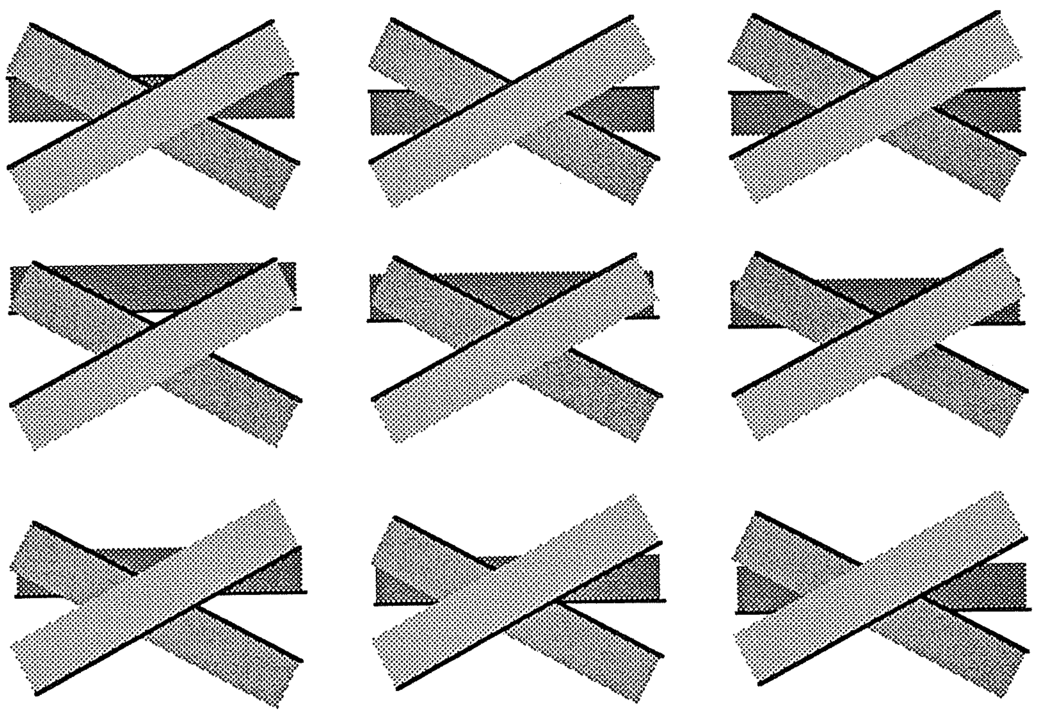


Figure 3-13. Edge--edge-edge events. In each row, the center figure shows the event and the left and right figures show the aspect on either side.

These are the only kinds of edge-edge-edge occlusion that occur. Regions cannot split or merge in the case of edge-edge-edge occlusion because the merging of regions requires that the occluding faces separate. That implies a that a vertex and an edge appear to intersect at that image point or that parallel lines appear to overlap in the image. The latter case is handled as two instances of edge-vertex occlusion.

The boundaries in viewpoint space generated by edge-edge-edge events occur at viewpoints where three unconnected edges appear to intersect in a single point. In order to find the viewing directions in which three edges appear to intersect in a single point, let the endpoints of the three edges be $\mathbf{p}_{11}$, $\mathbf{p}_{12}$, $\mathbf{p}_{21}$, $\mathbf{p}_{22}$, $\mathbf{p}_{31}$, and $\mathbf{p}_{32}$ respectively, and pick a point on one of the edges, say

$$\mathbf{p} = \mathbf{p}_{11} + t\,(\mathbf{p}_{12} - \mathbf{p}_{11}),\quad 0 \le t \le 1 \tag{3-1}$$

The other two edges appear to intersect in some viewing direction **d** at **p** whenever the planes defined by **p** and each of the other two edges intersect in a line with direction **d**. That is, the other two edges appear to intersect at **p** from a viewing direction given by the intersection of the planes defined by **p**, $\mathbf{p}_{21}$, $\mathbf{p}_{22}$, and **p**, $\mathbf{p}_{31}$, $\mathbf{p}_{32}$. Normals to these planes are given by

$$(\mathbf{p} - \mathbf{p}_{21}) \times (\mathbf{p}_{21} - \mathbf{p}_{22})$$

$$(\mathbf{p} - \mathbf{p}_{31}) \times (\mathbf{p}_{31} - \mathbf{p}_{32})$$

and the intersection of these two planes has the direction

$$\mathbf{d} = [(\mathbf{p}-\mathbf{p}_{21}) \times (\mathbf{p}_{21}-\mathbf{p}_{22})] \times [(\mathbf{p}-\mathbf{p}_{31}) \times (\mathbf{p}_{31}-\mathbf{p}_{32})] \qquad (3\text{-}2)$$

Therefore **d** is a viewing direction in which the three edges appear to intersect at **p**. Note that **d** is a quadratic function of the parameter *t*. Thus viewing directions do not form an arc of a great circle in viewpoint space—rather, they form a quadratic curve when projected onto a plane. In Eq. (3-2) **d** is given in Cartesian coordinates. In order to transform it to the (θ,ϕ) notation, we can use a Cartesian-to-spherical coordinate transformation:

$$\theta = \tan^{-1} (d_x/d_z) \qquad (3\text{-}3)$$

$$\phi = -\sin^{-1} (d_y/\sqrt{d_x^2 + d_y^2 + d_z^2}) \qquad (3\text{-}4)$$

There are $O(\mathbf{n}^3)$ sets of three edges in a polyhedron, so there are $O(\mathbf{n}^3)$ such curves in viewpoint space.

Note that horizon and edge-vertex occlusion boundaries can be considered a special case of edge-edge-edge occlusion. When two of the edges of edge-edge-edge occlusion meet at a vertex of the object, the result is edge-vertex occlusion. When all three of the edges lie on the same face of the object, the result is that the viewing direction lies in the plane containing the face. In that case the viewing direction is on the boundary of occlusion caused by the face turning away from the viewer.

We claim that the occlusion boundaries listed above represent a complete catalog of events that change the topology of an image. If the topology of the image is to change with a small change in viewpoint, then some region must appear or disappear, regions split or merge, or regions gain or lose edges. If one of these things happens, it must start at some point in the image. The only way that such a change can happen is if at least three edges (or two parallel edges) meet at that point in the image, and we have listed all the unique ways in which three edges can meet. If more than three edges meet at the point, then either they are degenerate and some subset of three generates the same event, or they are general and hence do not generate an event with viewpoint space extent; i.e., the lines appear to intersect at only a single viewpoint. Parallel edges that overlap in an image can be treated as two edge-vertex pairs.

3-5.1.2. An Upper Bound

We have seen that the boundary points of occlusion are viewing directions where three object edges appear to intersect in a single point, and there are $O(\mathbf{n}^3)$ curves of such points. $O(\mathbf{n}^3)$ quadratic curves on the sphere intersect in $O(\mathbf{n}^6)$ points. The resulting subdivision of viewpoint space is planar, so it also has $O(\mathbf{n}^6)$

edges and regions. Therefore the VSP and the aspect graph both have maximum size $O(\mathbf{n}^6)$.

3-5.1.3. A Lower Bound

We have argued that there are $O(\mathbf{n}^3)$ curves in viewpoint space potentially bounding regions of visibility and that there are $O(\mathbf{n}^6)$ intersection points of these curves, so the VSP has size $O(\mathbf{n}^6)$. In fact, these bounds are tight. We show this by presenting an example family of polyhedra with **n** faces that has an aspect graph (and hence a VSP) of size $\Omega(\mathbf{n}^6)$.*

Consider two grids of **m** strips each, with the strips close together. In front of these grids are two screens, each with **m** slits. The two screens and grids are arranged as shown in Figure 3-14. Note that the grid edges are not quite parallel to the screen edges.

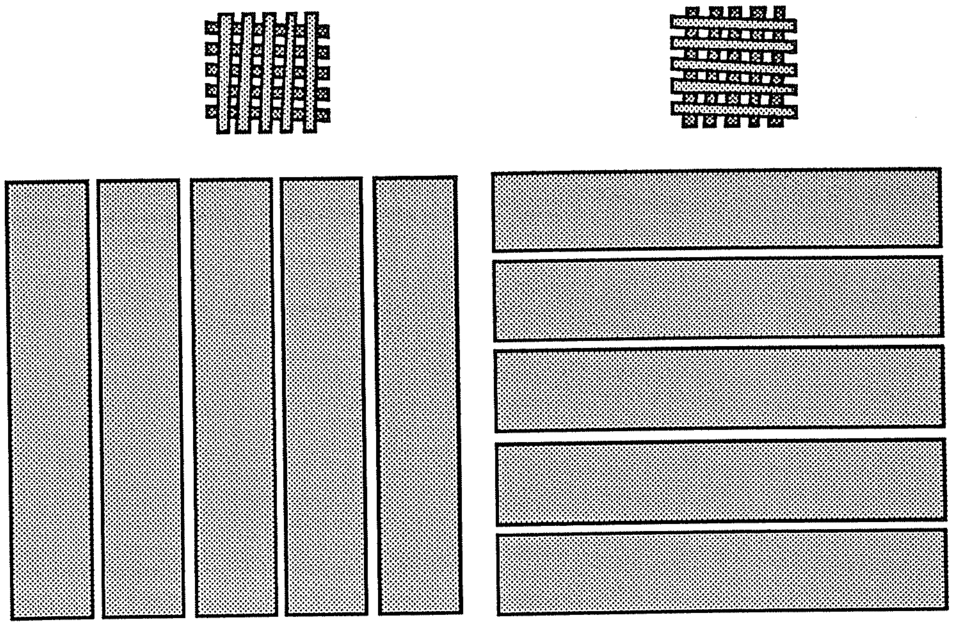


Figure 3-14. Two grids behind two screens. Note that the grids are not quite parallel to the screens.

In a typical view of the grids behind the screens, parts of the grids are visible through only one slit of each screen. Furthermore, only a small part of the grid is visible through the slit (see Figure 3-15).

* This example is due to John Canny [1987].

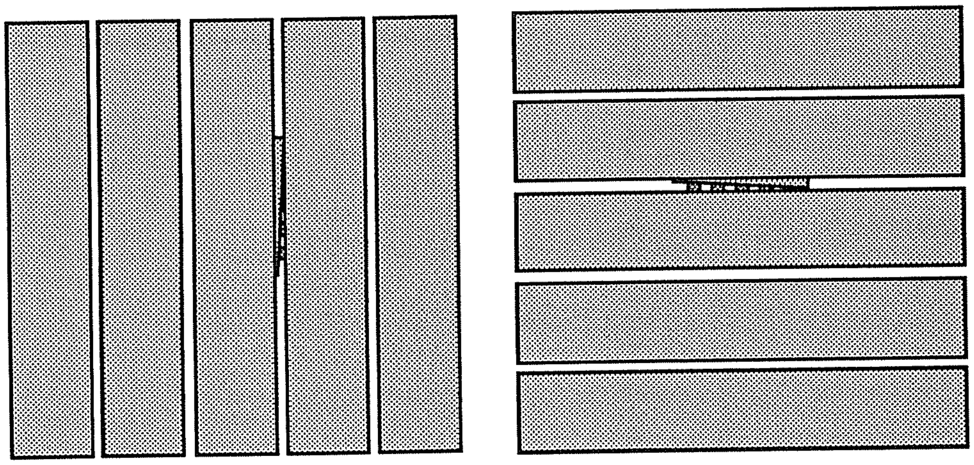


Figure 3-15. A typical view of the two grids (seen simultaneously).

The part of each grid that is visible behind the vertical screen is a portion of a nearly-vertical grid element and between 0 and **m** nearly-horizontal grid elements. The view through a slit of the horizontal grid is symmetrical. Figure 3-16 contains close-up views of the parts of the grids visible through the two slits.

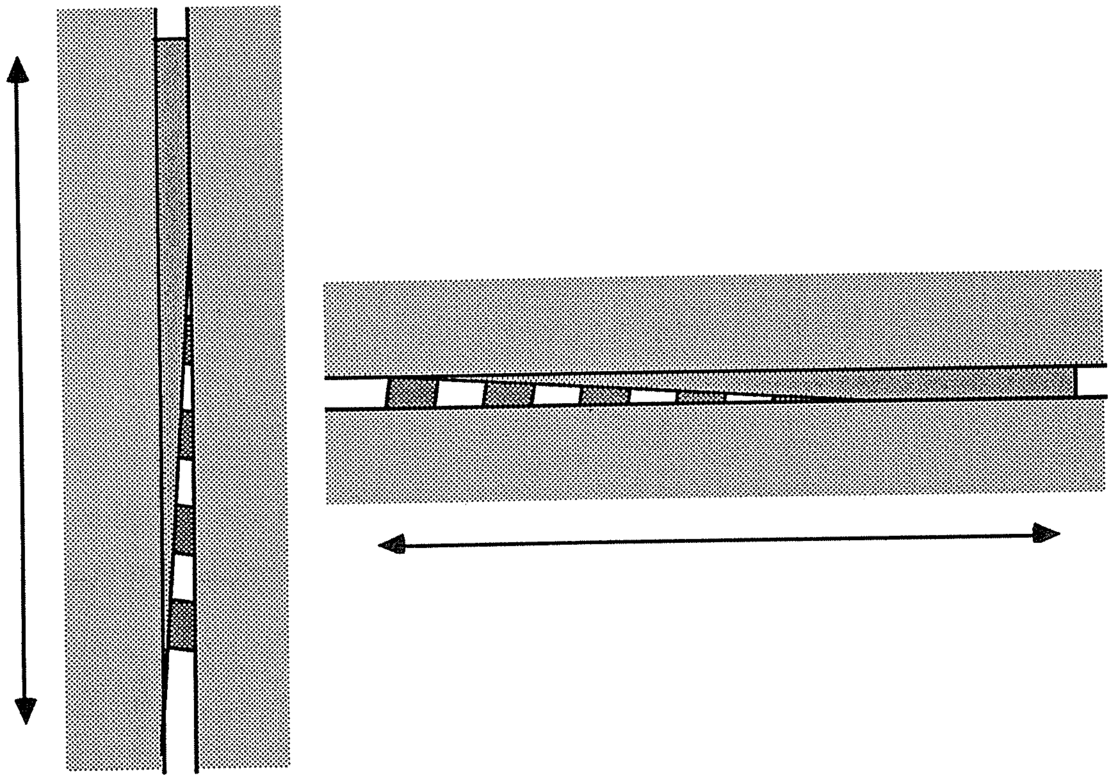


Figure 3-16. Close-up views through the slits.

Note that changing the viewing direction along the horizontal arrow causes the **m** nearly horizontal faces to disappear from view in the vertical slit and reappear one by one, generating a new visual event each time. This disappearing and reappearing of the horizontal faces in back occurs **m** times, so that the view through this slit requires $\Omega(\mathbf{m}^2)$ vertices in the aspect graph. There are **m** vertical slits, requiring $\Omega(\mathbf{m}^2)$ boundaries each, for a total of $\Omega(\mathbf{m}^3)$ boundaries. These boundaries are parallel straight lines (arcs of great circles) in viewpoint space. Note also that the view through a vertical slit does not change when the viewing direction is changed parallel to the vertical arrow.

The views through the horizontal slits exhibit the same behavior: they cause $\Omega(\mathbf{m}^3)$ straight boundaries in viewpoint space. However, the boundaries are orthogonal to the other $\Omega(\mathbf{m}^3)$ boundaries, so there are $\Omega(\mathbf{m}^6)$ intersection points of these boundaries. Therefore the VSP and the aspect graph have size $\Omega(\mathbf{m}^6)$.

3-5.1.4. An Algorithm

An algorithm for constructing the VSP in the case of convex polyhedra is relatively straightforward. The VSP is the partition of the sphere generated by great circles corresponding to the faces of the polyhedron. In the case of non-convex polyhedra this approach does not work. A face may not be visible from all viewpoints in front of it because it is partially occluded from some viewpoints by other faces. However, the types of boundaries of regions of the VSP (events) were listed above, and the VSP can be computed from them.

3-5.1.4.1. A Naive Approach

Since every boundary of a region of constant aspect in viewpoint space is of the form of the edge-edge-edge boundaries defined above, a naive approach is to find *every* boundary in viewpoint space generated by every set of three object edges, taking those edges in all possible orders. Eq. (3-1) gives a potential boundary of visibility in viewpoint space for such a set of three edges. We can draw all $O(\mathbf{n}^3)$ of these boundaries on the sphere and then find the subdivision of the sphere that they generate.

The subdivision of viewpoint space generated by these boundaries has size $O(\mathbf{n}^6)$ and can be constructed in $O(\mathbf{n}^6 \log \mathbf{n})$ time using the algorithm given in Section 3-5.1.4.2. This subdivision is a refinement of the VSP (i.e. has all the vertices and edges of the VSP and more) since the boundaries are only *potential* boundaries of constant aspect; not all of them actually represent changes in visibility. For example, some of the potential events may not be visible in any image. It remains to remove the boundaries that are not actual aspect boundaries.

For any boundary in viewpoint space, we know the edges that gave rise to the boundary and the image point at which the region appeared or disappeared. Thus, computing the faces visible immediately around the image point in question from viewpoints immediately adjacent to the boundary suffices to determine whether the aspect actually changes at that boundary: if the two sets of visible faces are equal, the aspect does not change, and vice versa. We then merge adjacent regions if the same set of faces is visible from both regions. The result is the VSP, and its dual is the aspect graph.

This algorithm can be executed in $O(\mathbf{n}^7)$ time, which is somewhat worse than the worst case size of the aspect graph, $O(\mathbf{n}^6)$. A more serious problem with this algorithm is that its best case time is also $\Omega(\mathbf{n}^7)$. For example, the VSP for a convex polyhedron has size $O(\mathbf{n}^2)$ but this algorithm would still take time $\Omega(\mathbf{n}^7)$. Since the worst case is very large and most aspect graphs are not as large as the worst case, an algorithm with better behavior for simple polyhedra would be much preferable.

The size of the subdivision generated by **m** curves is $O(\mathbf{m}^2)$, so in order to construct the aspect graph more efficiently we must compute the actual boundaries of aspect *before* generating the subdivision. Then if the actual number of aspect boundaries is smaller than $\Theta(\mathbf{n}^3)$, the runtime of the algorithm will be much less than $\Theta(\mathbf{n}^6 \log \mathbf{n})$. In order to do that, we make use of the aspect representation for the object. Since the asp represents all of the visual events for a polyhedron, it is easy to find the events from the asp. This approach is described in the next section.

3-5.1.4.2. Using the Asp to Construct the VSP

All boundaries of image regions are represented as a function of viewpoint by faces in the asp, so in particular every event that changes the topology of the image is represented in the asp. The asp faces corresponding to events are edges. In the definition of visual events given, all asp edges also correspond to events. Thus, to find all events we construct the asp for the polyhedron and project the asp edges into viewpoint space. In the alternate definitions of events, not all asp edges correspond to events; some correspond to changes that do not result in a topologically-distinct image. In that case, only the asp edges corresponding to events should be considered in constructing the VSP. For example, if the definition of an event requires that the set of visible image regions changes, the events listed in Section 3-5.1.1 that only change the number of edges in regions should not be considered. The remainder of the algorithm is the same.

The asp edges corresponding to events must be projected into viewpoint space, which is done by solving for viewpoint as a function of a single parameter. The equation for an asp 1-surface is given in this form in Eq. 2-16. We must also be able to find intersections of these curves in viewpoint space. Eq. 2-16 is a vector equation of the form

$$\mathbf{V}' = \mathbf{a}\, s^2 + \mathbf{b}\, s + \mathbf{c}$$

The intersection point of two such curves is the solution of the vector equation

$$\mathbf{a}_1 s_1^2 + \mathbf{b}_1 s_1 + \mathbf{c}_1 = \mathbf{a}_2 s_2^2 + \mathbf{b}_2 s_2 + \mathbf{c}_2$$

The solutions of this equation can be found with the quadratic and quartic equations. There are at most four intersection points of the two curves.

It remains to construct the subdivision of viewpoint space generated by these events. We call the cells of the subdivision *regions*, the boundaries of the regions *edges*, and the intersection points of the edges *vertices*. We will use a data structure for the subdivision with nodes for regions, edges, and vertices, and pointers between nodes corresponding to adjacent regions and edges or edges and vertices. At each node corresponding to an edge we store the constants of the curve on which the edge lies.

The edge-vertex structure of the subdivision generated by **m** curves on the sphere is constructed incrementally, starting with an empty subdivision and adding curves one by one. Adding a curve **c** to the subdivision involves finding the intersection points of **c** with every curve already in the subdivision and inserting them into the subdivision. **c** will intersect each curve already in the subdivision at most four times. Each intersection point is added to the subdivision by splitting the two edges that cross at an intersection point into four edges that meet at that point. The edge on which the intersection point lies can be found in O(log **n**) time with binary search.

After all of the curves have been added to the subdivision, the result is a structure of edges and vertices. The regions can be added with a graph-search algorithm in linear time in the size of the edge-vertex structure. Since each intersection point is found in O(log **m**) time, a subdivision generated by **m** curves is constructed in O($\mathbf{m}^2$ log **m**) time.

The asp is constructed in O($\mathbf{n}^5$) time (see Chapter 2) and results in O($\mathbf{n}^3$) boundaries in viewpoint space. The subdivision can be constructed in O($\mathbf{n}^6$ log **n**) time, so the time to construct the VSP is O($\mathbf{n}^6$ log **n**). The construction time is dominated by the time to find the subdivision unless the polyhedron is highly degenerate and does not generate many events. That can occur when most of its edges are parallel or meet at a single vertex. The subdivision algorithm takes time O(**m** log **m**) for an output of size **m** for any subdivision that is not highly degenerate, so the runtime of the VSP construction algorithm is nearly optimal in the sense that its runtime is within a log factor of the output size for polyhedra that are not highly degenerate.

This algorithm is slightly more efficient than the naive algorithm for constructing the VSP in the worst case, O($\mathbf{n}^6$ log **n**) vs. Θ($\mathbf{n}^7$). It is much more efficient than the brute-force algorithm for objects that are not as visually complex as the worst case. For example, constructing the asp for a convex object takes O($\mathbf{n}^2$) time using the algorithm given, and finding the VSP generated by the O(**n**) events takes O($\mathbf{n}^2$ log **n**) time. The naive algorithm requires Θ($\mathbf{n}^7$) time even in this case.

The aspect graph is the dual of the region-boundary structure of the VSP. We find it by finding the dual of the VSP, as in the convex case. The aspect graph has the same maximum size as the VSP, $\Theta(\mathbf{n}^6)$. The aspect graph has a vertex for every distinct aspect of the polyhedron, but as we have defined it, the vertices of the aspect graph contain no information about the particular aspect that they represent. In order to characterize the aspect one can store an image of the object for one of the views in the viewing region of the VSP. This requires $O(\mathbf{n}^2)$ space for each aspect and can be computed in $O(\mathbf{n}^2)$ time with a hidden-line removal algorithm, so the aspect graph with images at each vertex requires space and time $O(\mathbf{n}^8)$ to construct.

3-5.2. Perspective Case

Under the perspective model, the viewpoint is unrestricted, so viewpoint space is R^3. Therefore the VSP is a partition of R^3. It has cells corresponding to volumes of viewpoint space of constant aspect. Since we are modelling "looking at" an object that is near the origin, the viewing direction is again the direction from the viewpoint to the origin.

We also no longer assume that the whole object is always in front of the viewer as in the orthographic model. For example, in modelling a house under the orthographic model, the only viewpoints allowed are a sphere of viewpoints outside of the house. However, in the perspective model viewpoint space is R^3, so the viewpoint can be outside of the house, in the living room, or inside a closet.

3-5.2.1. Occlusion under Perspective Projection

The same visual events that occur in the orthographic model (horizon effect, edge-vertex, and edge-edge-edge) also occur in the perspective model. However, since viewpoint space is R^3, the boundary generated in viewpoint space in each case is a surface in R^3 rather than a curve on the sphere. In the orthographic model, a face turning away from the viewer generates a visibility boundary that is an arc of a great circle on the sphere. In the perspective model, the boundary of visibility is a plane in viewpoint space, specifically, the plane containing the face. This is true because a face "turns away from the viewer" whenever the viewpoint drops below the plane containing the face.

Edge-vertex occlusion boundaries (see Figure 3-11 and 3-12) are also parts of a plane. The plane is defined by the vertex and the edge involved in the visual event. The lines bounding the section of the plane corresponding to the occlusion boundary are the lines defined by the vertex and each of the endpoints of the edge.

Edge-edge-edge occlusion boundaries (see Figure 3-13) are the general sort of occlusion boundaries. In the orthographic model they are curves on the sphere; in the perspective model they are surfaces in R^3. Thus there is a surface of viewpoints from which three edges appear to intersect in a point. The viewing directions in which the three edges appear to intersect in a single image point is a line of viewpoints parallel to the viewing direction of the orthographic case (Eq. 2-16). The line passes through a point on an object edge at which all three edges appear to intersect, so the equation for a viewpoint from which the three edges appear to intersect in a single point is

$$\mathbf{V} = \mathbf{p}_1 + s\,\mathbf{a}_1 + r\,[((\mathbf{p}_1 + s\,\mathbf{a}_1 - \mathbf{p}_2) \times \mathbf{a}_2) \times ((\mathbf{p}_1 + s\,\mathbf{a}_1 - \mathbf{p}_3) \times \mathbf{a}_3)] \qquad (3\text{-}5)$$

3-5.2.2. Upper Bound

The surfaces in R^3 that bound the regions of constant aspect are generated by visual events involving triples of object edges. There are $O(\mathbf{n}^3)$ triples of edges and hence $O(\mathbf{n}^3)$ such surfaces for a polyhedron of size $\mathbf{n}$. Since the surfaces are algebraic, any pair has a constant number of curves of intersection and any three have a constant number of points of intersection. Thus the $O(\mathbf{n}^3)$ surfaces have $O(\mathbf{n}^9)$ intersection points.

On any one of the surfaces there can be at most $O(\mathbf{n}^6)$ edges and faces because the intersection of that surface with the other surfaces results in a 2-D subdivision of that surface by $O(\mathbf{n}^3)$ curves. Summing for all surfaces, a total of $O(\mathbf{n}^9)$ vertices, edges, and faces bound cells of viewpoint space. Since a cell must have bounding faces and a face can bound at most two cells, there are $O(\mathbf{n}^9)$ cells. Thus, the VSP has size $O(\mathbf{n}^9)$.

3-5.2.3. Lower Bound

In this section we show that $O(\mathbf{n}^9)$ is a tight bound on the maximum size of the VSP and aspect graph. We present a polyhedral scene that has a VSP of size $\Omega(\mathbf{n}^9)$. The example is similar to the lower bound example in the orthographic case (Section 3-5.1.3) except that changing the viewpoint in any of three orthogonal directions changes the aspect (see Figure 3-17).

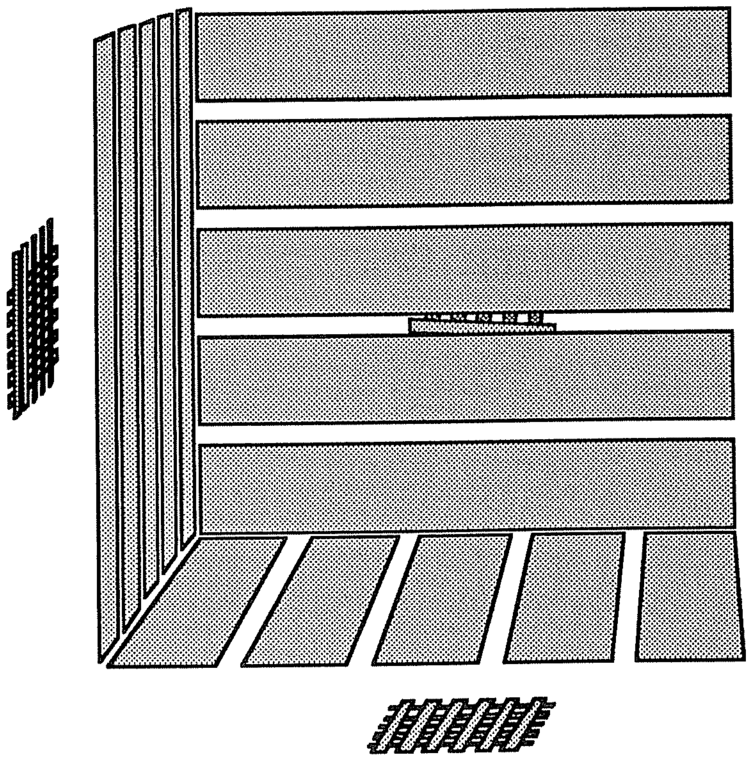


Figure 3-17. Polyhedra with VSP of size $\Omega(n^9)$.

Figure 3-17 presents a polyhedral scene with three screens on three adjacent sides of a cube. There are also three grids, each one a distance behind one of the screens. We will show that there are $\Omega(\mathbf{n}^9)$ different aspects from viewpoints inside the cube.

From any viewpoint inside the cube, some subset of faces of each grid is visible through one of the slits. In a manner similar to the orthographic example of Section 3-5.1.3, changing the viewpoint parallel to one of the edges of the cube changes the faces of one grid through one screen, but it does not affect the view of the other two grids. Each grid generates $\Omega(\mathbf{n}^2)$ boundaries of visibility through each slit, or $\Omega(\mathbf{n}^3)$ boundaries of visibility through the whole screen. All of the boundaries are parallel planes, so the $\Omega(\mathbf{n}^3)$ planes corresponding to each screen/grid pair intersect in $\Omega(\mathbf{n}^9)$ points. Thus the VSP and the aspect graph have size $\Omega(\mathbf{n}^9)$.

Note that we could just as well have used orthographic projection in constructing this example. However, in our orthographic model we define viewpoint space to be the sphere. This example would suffice as a $\Omega(\mathbf{n}^9)$ lower bound under orthographic projection as well if viewpoint space is R^3.

3-5.2.4. An Algorithm

Since the VSP is a partition of R^3 under the perspective model, in order to construct it we must be able to calculate the surfaces in viewpoint space that bound the aspects. These surfaces are given by Eq. (3-5) for sets of three object edges in all orders. As in the orthographic case, we can calculate the VSP using a naive algorithm: we construct all possible boundaries of visibility using Eq. (3-5) for all sets of three object edges in all orders. We find the subdivision of viewpoint space that these boundaries generate; the result is the VSP.

A subdivision of R^3 by $\mathbf{m}$ algebraic surfaces has size $O(\mathbf{m}^3)$, so the $O(\mathbf{n}^3)$ boundaries in viewpoint space generate a subdivision of size $O(\mathbf{n}^9)$ with a 3-D viewpoint space. The subdivision can be constructed in $O(\mathbf{n}^9 \log \mathbf{n})$ time using the algorithm given in Section 3-5.2.4.2. It remains to test each face $\mathbf{f}$ that separates two cells of the subdivision to determine whether the aspect actually changes at $\mathbf{f}$. We can test whether the aspect actually changes by finding the appearance of the image immediately around the image point where the event occurs that gave rise $\mathbf{f}$. If images near the event are the same from viewpoints immediately on either side of $\mathbf{f}$, that face should be removed from the VSP and the cells on each side merged. This test can be performed in $O(\mathbf{n})$ time for each face of the VSP, so using this naive algorithm the VSP can be constructed in $O(\mathbf{n}^{10})$ time.

Unfortunately, as in the orthographic case, the best-case and worst-case times are the same for this naive algorithm: it requires time $\Omega(\mathbf{n}^{10})$ even for simple objects. An algorithm will perform better on the average if it finds the exact set of boundaries in viewpoint space *before* finding the subdivision that they generate. It is important to keep the number of boundaries in viewpoint space as small as possible before finding the subdivision that they generate because $\mathbf{m}$ boundaries that are not highly degenerate form a subdivision of size $\Theta(\mathbf{m}^3)$. In order to compute the exact visibility boundaries for each face of the polyhedron, we again use the aspect representation, this time constructed using perspective projection.

3-5.2.4.1. Using the Asp to Construct the VSP

As in the orthographic case, all boundaries of VSP regions are represented as a function of viewpoint by faces in the asp. In the perspective case, events are the projections of 2-faces of the asp into viewpoint space. Thus, we find the events for a polyhedron by constructing the asp for the polyhedron and projecting the 2-faces into viewpoint space. Events found in this manner are visible, since they occur in the asp. It remains to show how to project the 2-faces into viewpoint space and find the subdivision of viewpoint space they generate.

Projecting the 2-faces into viewpoint space is done by solving the equation for a 2-surface for viewpoint as a function of two parameters. The equation for the 2-surface is already expressed in this form in Eq. (3-5). The form of Eq. (3-5) is

$$\mathbf{V} = \mathbf{a}_1 + s\,\mathbf{a}_2 + r\,[\,s^2\,\mathbf{a}_3 + s\,\mathbf{a}_4 + \mathbf{a}_5]$$

The intersection of two such surfaces is a curve of the form

$$\begin{aligned} &\mathbf{a}_1 + s_1\,\mathbf{a}_2 + r_1\,[\,s_1^2\,\mathbf{a}_3 + s_1\,\mathbf{a}_4 + \mathbf{a}_5] \\ &\quad = \mathbf{a}_6 + s_2\,\mathbf{a}_7 + r_2\,[\,s_2^2\,\mathbf{a}_8 + s_2\,\mathbf{a}_9 + \mathbf{a}_{10}] \end{aligned} \qquad (3\text{-}6)$$

This equation is satisfied for some values of r_1 and r_2 whenever three vectors are coplanar: the two viewing directions $[s_1^2\,\mathbf{a}_3 + s_1\,\mathbf{a}_4 + \mathbf{a}_5]$ and $[s_2^2\,\mathbf{a}_8 + s_2\,\mathbf{a}_9 + \mathbf{a}_{10}]$ and the vector between the two object points at which the event occurs, $\mathbf{a}_1 + s_1\,\mathbf{a}_2 - \mathbf{a}_6 - s_2\,\mathbf{a}_7$. Therefore Eq. (3-6) is satisfied when the triple cross product of these three vectors is zero. This yields an equation that is cubic in s_1 and s_2,, and it can be solved for s_2 in terms of s_1 with the cubic equation. The intersection of three surfaces of the form of Eq. (3-5) yields two algebraic equations for s_2 in terms of s_1; finding intersection points involves setting the equations equal and finding zeros using numerical methods.

It remains to construct the subdivision of viewpoint space generated by these events. We will call the volumes in viewpoint space of the subdivision and 2-, 1-, and 0-dimensional boundaries *cells*, *faces*, *edges*, and *vertices*, respectively. We use a data structure similar to that of the subdivision of the sphere, with the addition of nodes for cells and links from each cell to the faces bounding it. For each face we store the constants defining the surface on which it lies.

The subdivision generated by **m** surfaces in R^3 is constructed incrementally by adding surfaces to a partial subdivision. A surface is added to the subdivision by finding the intersections of that surface with every other surface. The intersection curves are added to the 2-D subdivisions on each surface in the same way that the 2-D subdivision was constructed in the orthographic case. After the edge-vertex structure of the subdivision has been constructed, the faces on each surface are found in the same manner as in the orthographic case, and the cells are added in O(**n**) time using a graph-traversal algorithm. Since each intersection point is found in O(log **m**) time, a subdivision of **m** curves is constructed in O($\mathbf{m}^3$ log **m**) time.

The asp is constructed in O($\mathbf{n}^5$) time (see Chapter 2) and results in O($\mathbf{n}^3$) boundaries in viewpoint space. Therefore the subdivision can be constructed in O($\mathbf{n}^9$ log **n**) time, so the time to construct the VSP is O($\mathbf{n}^9$ log **n**). The time complexity of constructing the VSP is dominated by the time to find the subdivision unless the polyhedron is highly degenerate and does not generate many events. The subdivision algorithm takes time O(**m** log **m**) for an output size of **m** for any subdivision that is not highly degenerate, so the runtime of the VSP construction algorithm is nearly optimal in the sense that it is within a factor of O(log **m**) of the output size for polyhedra that are not highly degenerate.

The algorithm for constructing the VSP is more efficient than the naive algorithm in all cases: the runtimes are O($\mathbf{n}^9$ log **n**) and Θ($\mathbf{n}^{10}$) respectively. However, the asp algorithm is far more efficient in the case where the polyhedron is visually simple. For example, for a convex polyhedron the naive algorithm requires time Θ($\mathbf{n}^{10}$), but the asp algorithm takes time O($\mathbf{n}^3$ log **n**).

The aspect graph, which is the dual of the cell-face structure, is constructed in linear time in the size of the VSP in the same manner as in the convex case. The aspect from a viewpoint requires $O(\mathbf{n}^2)$ time to compute and $O(\mathbf{n}^2)$ space to represent, so storing the aspect at each vertex of the aspect graph in the perspective case requires time and space $O(\mathbf{n}^{11})$.

3-6. Concluding Remarks

In this chapter we analyzed the visibility and occlusion of faces of a polyhedron over all viewpoints. We showed how to calculate boundaries of visibility and regions of constant aspect in viewpoint space. We did this by listing the ways in which aspect can change. The aspect changes when a region in the image appears or disappears, or when regions merge or split.

We used two tools for characterizing aspect over all viewpoints: the viewpoint space partition and the aspect graph. We presented tight bounds on the maximum size of the VSP and aspect graph in the convex and non-convex cases, using two models: orthographic projection and a 2-D spherical space of viewpoints, and perspective projection where viewpoint space is R^3. Thus we bounded the maximum number of topologically-distinct views of a polyhedral object in those cases. Under the orthographic model the maximum size is $\Theta(\mathbf{n}^2)$ in the convex case and $\Theta(\mathbf{n}^6)$ in the non-convex case. Under the perspective model the maximum size is $\Theta(\mathbf{n}^3)$ in the convex case and $\Theta(\mathbf{n}^9)$ in the non-convex case. The difference in size between the two models is due to the difference in dimensionality of the viewpoint space in each model, not the use of orthographic or perspective projection.

We also presented algorithms for constructing the VSP and the aspect graph for convex and non-convex polyhedra, under orthographic and perspective projection. The runtimes of these algorithms are summarized in Table 3-2. The algorithms involved determining the visibility of each face of the polyhedron as a function of viewpoint, which is represented using the asp. The asp has an edge for each VSP region boundary, so constructing the VSP is done by finding the projections of the asp edges on viewpoint space and computing the subdivision of viewpoint space that they generate. The algorithms for convex polyhedra run in time that is worst-case optimal and in fact optimal for all polyhedra that are not highly degenerate. The algorithms for non-convex polyhedra run in nearly optimal time in the sense that the runtime is within an $O(\log \mathbf{m})$ factor of output size $\mathbf{m}$ for any polyhedron that is not highly degenerate.

One motivation for constructing the aspect graph is to find the topologically-distinct views of an object for the purposes of object recognition. However, we have shown that the number of views of a non-convex polyhedron can be extremely large, $O(\mathbf{n}^6)$ in the orthographic case and $O(\mathbf{n}^9)$ in the perspective case. Therefore, algorithms for object recognition that work on relatively large and complex objects probably cannot use all topologically-distinct views.

An interesting extension to the algorithm presented in this chapter would be to construct an approximation to the VSP that has smaller size but still captures the most important changes in visibility. One approach to this is to construct a hierarchical VSP by constructing the VSP for each level of a hierarchical representation of the object. It would also be interesting to construct the VSP for objects with some curved faces, such as cylinders, or to define and construct the VSP procedurally, in a manner similar to a constructive solid geometry representation.

It would also be interesting to define and construct the VSP and the aspect graph under different viewing models. In the given models, viewpoint space is partitioned according to the visibility of edges in the image. However, other notions of visibility may be useful for some problems. For example, a new viewing model might be defined in such a way that only occluding edges are visible in an image. Perhaps they are found as discontinuities in a range image. The model would then match the sensor output more closely, and the storage space would also be reduced since there are fewer occluding edges than polyhedral edges in an image.

Chapter 4: Fast Parallel Object Recognition Using a Representation of Appearance

4-1. Introduction

Currently, the model of 3-D object recognition that is the most influential is probably Marr's object-centered model [1982]. In Marr's model, early, bottom-up visual processing results in a representation called the "2 1/2-D sketch." This representation is in register with the image, and for each image point it contains information such as the distance, orientation, and other properties of the corresponding world point. From the 2 1/2-D sketch, an object-centered model of the object in the image is constructed, called the 3-D model. Finally, recognition proceeds as a process of matching volumes of space: the 3-D model is matched with object-centered representations of known objects.

There are some problems with the object-centered model of object recognition. Some of the steps are very difficult; for example, consistently assigning an object-centered frame of reference to an object requires recognizing an axis of symmetry or using some other method of insuring that the same frame of reference will be assigned to the same object each time. Constructing an accurate object-centered model of an object from an image is as difficult as recognition itself since parts of the object are hidden, unless objects are assumed to be generalized cylinders or other constraints on shape are assumed.

In addition, much of the processing required in this model of object recognition is inherently sequential. Constructing an object-centered representation from the 2 1/2-D sketch and matching that representation with a model appear to be time-consuming sequential tasks [Rosenfeld, 1987]. The 3-D model can be matched with separate objects in parallel, but it is difficult to see how the 3-D model can be constructed and matched with a single model object nearly as quickly as humans can recognize unexpected objects, namely, in around 100 neuron-firing times.

The multiview or characteristic view approach to 3-D object recognition has been suggested as an approach that may result in recognition of unexpected objects in time comparable to that which humans achieve [Rosenfeld 1987]. This viewer-centered approach is inspired by the idea that comparisons of simple property measures of images may suffice for recognition. These can be carried out in parallel, so recognition can be accomplished quickly. The main drawbacks of this approach are the conflicting goals of a sufficient representation for recognition, requiring many views of an object, and a reasonable usage of storage and processing time, requiring fewer views.

One approach that is sometimes taken in an attempt to strike a balance between a sufficient viewer-centered representation and reasonable resource requirements is to represent all topologically-distinct views of an object. In this approach a view is stored from every aspect in the aspect graph (see Chapter 3). Since the aspect graph has a vertex for every topologically-distinct view of an object, a representation of all topologically-distinct views guarantees that the image will match one of the stored views topologically. Still, such a representation cannot guarantee a geometric match between an image and a stored view, and it has not been demonstrated that a topological correspondence is sufficient for recognition. Additionally, there are $O(\mathbf{n}^2)$ topologically-distinct views of a convex polyhedron and $O(\mathbf{n}^6)$ of a general polyhedron with $\mathbf{n}$ faces, so such representations may require a large amount of storage.

In this chapter we show how to use the aspect representation for 3-D object recognition. Since the asp represents appearance, recognition can be accomplished with fast, parallel techniques. In addition, since the asp represents appearance of an object from all viewpoints, the problem for the characteristic view approach of representing appearance from only a discrete set of viewpoints is eliminated. Size remains an advantage for object-centered representations, but surprisingly the asp has much smaller size than the aspect graph. Thus, the aspect representation has some of the benefits of both the object-centered and characteristic view approaches to object recognition.

Recognizing 3-D objects with the asp as the object model involves identifying features in an image, finding features of the same type in the asp, determining viewpoints from which image features and model features look the same, and verifying the viewpoints by comparing nearby features of the asp with the image. The process of recognition can be speeded by constructing indexes to various types of features in the asp and ranking feature types according to the amount of processing they are likely to require and the likelihood that recognition will result. The algorithm is stated in terms of parallel image feature detectors and associative memory. With sufficient processors and associative memory, it achieves recognition quickly, in a constant number of steps. With fewer processors the performance degrades moderately.

Asp object recognition differs fundamentally from other common approaches. Since the asp represents appearance rather than the volume of space that an object occupies, matching image features with model features involves matching features of appearance with a representation of appearance rather than with a 3-D model. As a result, recognition can be accomplished quickly in parallel. A related characteristic of asp object recognition is that self-occlusion is explicitly represented, so features of appearance involving self-occlusion can be matched with features represented in the asp. Figure 4-1 contains many features either caused by occlusion or made to look different by occlusion. Examples are T-junctions and image regions not corresponding to any object faces.

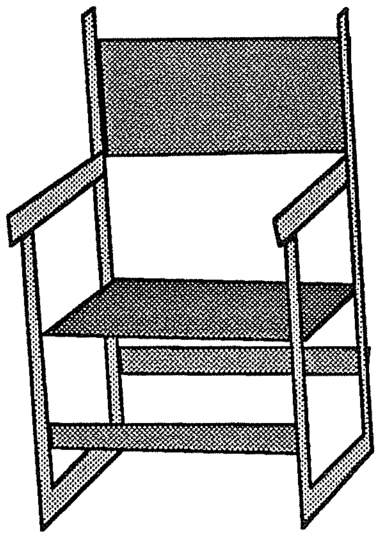


Figure 4-1. An object with many visual features resulting from occlusion.

Features caused by occlusion are not represented explicitly in object-centered representations, and therefore image features involving occlusion are not easily matched with the model. Asp object recognition also differs from the characteristic view approach, since appearance is represented from all viewpoints rather than a discrete set. In addition, the asp requires less storage.

Section 4-2 discusses the requirements of a fast object recognition system and different approaches to achieve fast recognition. Section 4-3 shows how to use the asp for object recognition. Section 4-4 compares asp object recognition with other methods, and Section 4-5 contains a discussion of the asp approach.

4-2. Fast 3-D Recognition of Unexpected Objects

3-D object recognition is the problem of matching an object in an image from an arbitrary viewpoint with one of many object models stored in memory. Humans are remarkably adept at the problem; we can recognize a single unexpected object in an image in around 100 neuron-firing times. That is particularly remarkable since a conservative estimate of the number of objects a person can recognize is 100,000 [Tsotsos, 1988].

Clearly humans make use of massive parallelism in recognizing objects. For example, an image may be matched with each known object simultaneously. However, even the task of matching an image with a single model in 100 cycles seems daunting. Images contain on the order of 10^6 pixels, and common methods for operations such as contour-following and segmentation require linear time in the number of pixels involved, which is too slow for fast recognition [Rosenfeld, 1987]. The constraint of recognizing objects in around 100 cycles is very strong, and it implies a number of constraints on the system used for recognition.

4-2.1. Properties of a Fast Object Recognition System

Since humans do a very good job of recognizing objects quickly, they serve as a demonstration that the massive parallelism and the organization of the brain are sufficient for recognition. Thus, the techniques used by humans are sufficient for recognition, and the performance achieved is plausible. Since humans can recognize objects in around 100 cycles, no procedures are required for recognition that are sequential in the data that they access: it must be possible to complete any process in a small constant number of steps. Furthermore, the connectivity of processors is limited in practice—in humans, to about 1000, for example. Thus, it is desirable that algorithms in some sense have *local* processing requirements. An algorithm that requires input from arbitrary parts of an image would be difficult to implement in a highly parallel system with limited connectivity.

Clearly, it is also desirable to use as little storage as possible for fast recognition in an object representation. However, the brain has around 10^{11} neurons, with a total of around 10^{14} synapses, so if at least 100,000 objects are to be recognized and as much as 10% of the brain's neurons are used for visual memory, the amount of storage available for each may be considerable; perhaps as much as 10^6 to 10^8 units. Of course, much of a human's visual memory will be used for remembering the appearance of particular objects rather than of types of objects. But these figures are rough estimates, probably within an order of magnitude or two of fact. If we assume that humans use visual memory efficiently and if as much as 10^6 to 10^8 units are available per object, it appears that considerable space is available in humans for reducing the runtime of recognition through redundant representations and space-time trade-offs. Of course, space is not unlimited, and no system should use more space than necessary.

Another desirable property of an object recognition system is that it should work for the case in which the scene changes with time because of movement of the viewer or the object. It should be possible to "track" a moving object without having to re-recognize it in each frame. In fact, moving objects should be easier to recognize since the motion reveals the depth and 3-D orientation of parts.

4-2.2. Viewer-centered Object Recognition

This section describes viewer-centered object recognition and reviews related work. For a survey of general 3-D object recognition, see [Besl and Jain, 1985] or [Chin and Dyer, 1986]. Viewer-centered object-recognition schemes seek to make use of parallel processing and plentiful memory by reducing the problem of object recognition to the simpler problem of matching between the image and stored images. One can think of the process involved as one in which views stored in associative memory "activate" when a similar view appears in an image. Matching an image with a similar stored image is not an easy problem either, but the recognition process is simplified by moving some of the work that an object recognition system must do (namely, determining viewpoint) to "smart" memory.

Many different schemes have been proposed for multiview or characteristic view object recognition. Some schemes represent views from a fixed number of viewpoints, either evenly spaced around the sphere or from stable orientations, i.e. orientations in which an object would rest on a table. Lieberman [1979] computes silhouettes of objects from stable orientations, assuming that there are only a few. Wallace *et al.* [1980] compute properties of the silhouettes of aircraft from a fixed number of viewpoints. Fekete and Davis [1984] introduce the idea of *property spheres*. The property sphere for an object represents properties of the object (in this case the silhouette) from 320 discrete viewpoints around the sphere.

One problem with this type of approach is that there is no reasonable criterion stated for determining an appropriate number of views for an object or from which viewpoints the views should be taken. For some objects, a few views would suffice, and for others many views would probably be necessary, but the number is either fixed for all objects or adjusted manually. Another problem is that no criteria are stated for determining when an image and a stored view match closely enough. As a result, there may be fewer or more views stored than necessary, and matching cannot be guaranteed.

An approach used to deal with these problems is the use of the aspect graph, introduced by Koenderink and van Doorn [1979] (see Chapter 3). Since the aspect graph has a vertex for every topologically-distinct view of an object, views can be selected in such a way that any image is guaranteed to match a stored view topologically, and no more views are stored than necessary for topological matching.

Using these ideas, Chakravarty and Freeman [1982] assume that objects may appear in one of a relatively small number of stable positions, and they represent one view of the object for each aspect corresponding to a stable position. For other views in an aspect they use linear transformations of the given view. Korn and Dyer [1985] grow regions of feature equivalence, storing one representative view of each region. Castore [1983], Castore and Crawford [1984], and Crawford [1985] store views for every aspect in the aspect graph.

There are still some problems with this kind of approach. First, it is unlikely that any particular image of an object will exactly match one of the stored views; in fact, one can only guarantee *topological* correspondence to one of the stored views, and it is not clear that topological correspondence is sufficient for recognition. Furthermore, topological correspondence is achieved only by storing an image for every vertex in the aspect graph, which can be very large. In Chapter 3 it is shown that the maximum number of vertices in the aspect graph is $\Theta(\mathbf{n}^2)$ for convex objects and $\Theta(\mathbf{n}^6)$ for general objects with $\mathbf{n}$ faces under orthographic projection, and $\Theta(\mathbf{n}^3)$ and $\Theta(\mathbf{n}^9)$ under perspective projection. Since an image of such a scene can have size $\Theta(\mathbf{n}^2)$ in the non-convex case, the respective worst-case sizes of a representation storing an image from every aspect are $\Theta(\mathbf{n}^3)$ and $\Theta(\mathbf{n}^8)$ even under orthographic projection.

Another problem with this sort of representation is that there is no information relating the different views of the object; the algorithms could as well be processing views of different objects. It would be beneficial to be able to interpolate between views. It is possible to use a linear transformation of the edges visible in an aspect if the relation between aspect edges and object edges is known, but in existing techniques for aspect graph-based object recognition, viewpoint boundaries separating aspects are not stored. Requiring that the image match a stored view to within a linear transformation still is not as strong as a geometric match when the boundaries on that transformation are not known.

4-3. Using the Asp for 3-D Object Recognition

In using the aspect representation for 3-D model-based object recognition, the asp is the model of an object to which an image is compared. In this chapter we will use the orthographic viewing model, as described in Chapter 2. In this model, objects in the world are polyhedra, and edges of polyhedra are visible in images, perhaps as a result of an edge-finding sensor system. With this viewing model, the asp for a polyhedron represents its appearance in any image. Since the focus of this chapter is on the representation and use of apparent shape for object recognition, the data will be assumed to be error-free.

We use a structural feature-based approach to object recognition. That is, the models may be thought of as a set of features arranged in a particular graph structure, which is represented as the asp for the object. When the method we propose finds feature matches, features that are nearby in the image are tested for a consistent feature match. Thus, the computation is local and easily performed in parallel. The algorithm involves several steps: identifying features in the image, finding the corresponding features in the asp, and for each match determining the viewpoint that would map the image feature onto the asp feature. If there is such a viewpoint, we compare nearby features in the image with nearby features in the appearance of the object at that viewpoint to determine whether we have in fact found a match.

4-3.1. Representing Appearance for Object Recognition

The aspect representation for a polyhedron represents its appearance from all viewpoints. Visual features such as polygons, edges, and T-junctions are represented as a function of viewpoint as cells, facets, and ridges of the asp. Matching is a process of matching an image with a representation of what may appear in the image (see Figure 4-2).

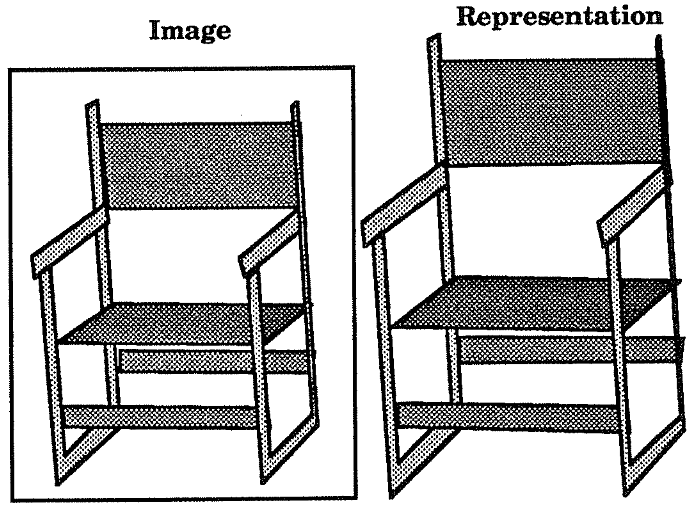


Figure 4-2. Using the asp for object recognition is a process of matching an image with a representation of what may appear in an image.

This is an important difference from object-centered representations. In an object-centered representation, a feature such as a polygon has no necessary correlation to a visible feature in the image. It may by completely visible or completely hidden, it may be partially visible as a polygon with fewer or more sides, or it may be visible in several disjoint regions of the image.

The problem with object-centered models is that visual features are represented implicitly, as the result of a series of computations such as hidden-line removal. With such a model, any sort of matching requires a significant amount of computation. It is therefore difficult to implement object-centered object recognition as a fast parallel algorithm. On the other hand, the asp represents appearance directly. In the asp, a cell corresponds directly to a visible polygon in the image. Thus, if a processor finds a polygon in the image and broadcasts that fact, a processor representing the polygon in memory can immediately activate, signifying a match.

Immediate matching can be approximated in the object-centered paradigm by assuming that the entire model is visible and there is no occlusion. Occlusion is treated as a noise process, a degradation of the image. In that case, what is represented in the model is assumed to be visible in the image, even though it is known to be wrong to some extent (see Figure 4-3).

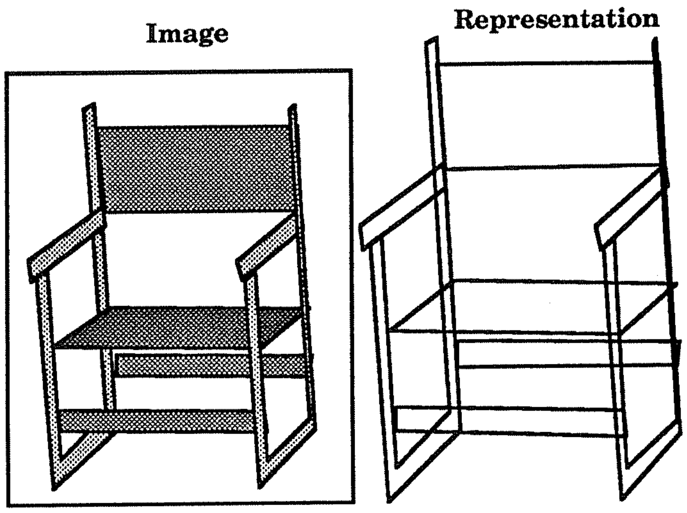


Figure 4-3. In object-centered object recognition, an image is matched with a representation of the volume of space an object fills, not a representation of appearance.

This approach is common among implementations of object-centered object recognition [Brooks, 1981; Lowe, 1986; Thompson and Mundy, 1987]. However, to attempt recognition with a model of something other than what is visible in the image is to attempt recognition with an unnecessary handicap. If sufficient memory and processing power is available, the correct approach to recognition is to model as closely as possible whatever the sensors will detect in the image.

4-3.2. Constructing Feature Indexes

In order to speed recognition, we can construct indexes to various features of appearance represented in the asp. The idea of using hierarchies of features for recognition is not new; for example, Ettinger describes the use of a library of parameterized model sub-parts for recognition [Ettinger, 1988]. The idea of finding locally consistent sets of features is also not new; Bolles and Cain [1982] and Bolles and Horaud [1986] show how to match locally consistent sets of image features and model features stored in a tree structure to hypothesize objects and orientations for recognition. We discuss using these ideas here to show that such an approach is also appropriate and beneficial for asp object recognition.

For example, suppose that one type of feature to be considered is a rectangle. All of the corresponding asp features (cells of aspect space with rectangular cross-section, in this case) can be located during the construction of the asp for an object, and an index can be created with a pointer to each such feature. When a rectangle is found in an image, we can quickly locate all of the rectangles in the asp using the index and compare the image rectangle to each of them. Such an index can be constructed for several different types of features. In addition, similar features in models of different objects can be combined to form an index into a large database of object models.

Another way to speed the recognition process is to compare image features with asp features in an order that is likely to speed recognition. Features found in the image that are uncommon and distinctive such as 37-sided polygons should be compared to asp features first since there are fewer possible matches to try. To determine the best order in which to select image features for matching with asp features, we determine the number of occurrences of each type of feature while constructing the indexes. We compare image features with asp features in order of increasing frequency of feature type. Thus, features that are less common and more likely to result in recognition quickly are tested first.

We will assume that the features are complex enough that a match between an image feature and an asp feature determines the viewpoint. (Note that some features can be matched in more than one way. For example, two rectangles can be matched four different ways.) Then we calculate the viewpoint, rotation, and translation of the object that satisfy the matching. In order to test the viewpoint calculated, we find adjacent faces of the asp that correspond to edges in the image that should be visible at that viewpoint, and we check the image to see if they are present.

Feature selection is important for any object recognition algorithm that hypothesizes matches between image and object features and tests whether the matches are correct. The ideal feature for such an algorithm has several properties:

- it is distinctive: it can be identified from a wide range of viewpoints, or at least classified as one of a small set of similar model features,
- a match between the image feature and the model feature determines the viewing parameters such as viewpoint, translation, and rotation, and
- incorrect matches between image and model features are infrequent.

With such a feature, we can find model features corresponding to an image feature quickly since we can identify it from a wide range of viewpoints. Hypothesizing that the image feature matches a model feature determines a viewpoint to test, and the number of viewpoints that are tested and turn out to be wrong is relatively small since incorrect matches between image and model features are infrequent.

There is a tradeoff between the complexity of features that feature detectors can identify and the number of errors in the results. In fact, Thompson and Mundy argue that the most complex sort of feature that should be used for recognition is a dihedral vertex [Thompson and Mundy, 1987]. However, in this chapter we are concerned with the representation and use for object recognition of shape as a function of viewpoint. In order to study that problem, we use a model of visibility in which edges of polyhedra are visible in the image without error. Thus, for our purposes we will use more complex features.

An example of a good feature to use for asp object recognition under our model of visibility is a polygon of a certain number of edges. We can construct indexes for three-, four-, five-sided polygons and so on, so that upon locating a polygon in the image we can quickly find polygons with the same number of sides in the asp. After hypothesizing a particular matching of vertices of an image polygon and an asp polygon, the viewpoint is determined, and if the polygon has four or more sides, the viewpoint is overconstrained, so that polygons will not often be judged to match incorrectly.

Since the asp represents self-occlusion information, features can be caused by occlusion as well as incidence. For example, T-junctions in an image are valid features to search for even though the two edges do not intersect in the object. Other examples of features that can be use for asp object recognition are special polygons such as rectangles, intersection points of several line segments, and sets of parallel lines.

In an image with edge elements identified and with a significant proportion of noise, features such as n-gons are very difficult for low-level, bottom up feature detectors, particularly in parallel. However, complex features such as these are made up of a local set of simpler features. For example, a quadrilateral is made up of simpler features such as edges and dihedral vertices. The more complex features may be found by adding a level to the hierarchy of the tree of features. When a simpler feature is found, the neighborhood may be searched for related features that form a more complex feature, in the manner of Bolles and Cain [1982].

4-3.3. Object Recognition

The object recognition process consists of a preprocessing phase and an on-line recognition phase. The preprocessing phase is the construction of object model database. A set of model feature types must be selected and indexes into the database for each type constructed. Then for each object to be recognized, the asp is constructed and added to the database. Each feature in the asp of the types being used is found and added to the appropriate feature index.

The on-line algorithm for recognition will be stated in terms of local, parallel computation. We assume that low-level processing results in a list of visible edges in the image. The algorithm for recognition will be described with the use of associative memory, which activates when something matching its contents is broadcast, and a set of processors detecting features in every location in the image. The goal of this model is to state the recognition algorithm in such a way that recognition occurs "quickly," i.e. within a constant number of steps, with enough processors. Such an algorithm can easily be simulated with fewer processors with a time penalty at worst inversely proportional to the fraction of the ideal number of processors used.

Recognition proceeds as follows: the image feature detectors look for the type of feature that they detect. There is one such set of processors for each location in the image, so all local features are found in constant time. Any features thus found are broadcast to the associative memory. This broadcast would be time-consuming if it were done sequentially. Instead, any feature detector that finds a feature broadcasts the fact to the associative memory for that particular type of feature. All such features are broadcast simultaneously, and the one with the highest confidence is received and tested first.

When the features have been broadcast to the associative memory, similar features in the knowledge base are found. When a match between an image feature and a model feature is found, the viewpoint at which the features have the same appearance is computed; the viewpoint from which a set of object points transforms into a set of image points can be found with a numerical technique such as that used by [Lowe, 1986]. The viewpoint at which the match occurs is broadcast to other nearby cells of associative memory for the same object. If all of the features are found in a local region and for the same viewpoint, then the processor for the object activates to signify that a match has been found.

With the ideal number of processors, the feature indexes do not speed recognition. However, with fewer than the ideal number, the indexes speed the process by allowing processors to quickly find model features matching given image features. The ordering of the testing of features by likelihood of recognition also speeds recognition only when fewer processors than the ideal number are available.

With a limited number of processors, the performance of the algorithm is not as good as human performance. However, since the algorithm can be simulated with fewer processor, the performance is still good. After finding features in the image, matching with features in the database proceeds quickly because of the feature indexes and the ordering of the feature tests.

With a processor for every object feature, the algorithm is faster; no searching through indexes is required. When features are broadcast, associative memory of a specific object feature activates upon discovering a match. When all of the processors for a part of an object activate with consistent viewpoint, the processor for the object activates and signifies that a match has been found. With enough processors, object recognition takes only a constant number of steps, a constant each for feature detection, broadcast of features found, associative memory matching, and consistency testing.

4-4. Comparisons with Other Methods of Object Recognition

3-D object-centered object recognition seeks to match an image of an object with a 3-D object-centered model. This is the most common approach to 3-D object recognition, and it has been around for a considerable length of time. Already in 1965 Roberts presents a system for matching edges in an image with the edges of 3-D object-centered models of simple polyhedra and identifying the polyhedron and determining its orientation [Roberts, 1965]. The work of Marr [1982] has been influential in object recognition, and it is strongly oriented toward object-centered models for object recognition. Marr suggests that low-level vision results in the *primal sketch*, a ~~representation in register with the image that contains edge and region information.~~ The next step in object recognition is the construction of the *2 1/2-D sketch*, another representation in register with the image that contains depth and orientation information. Finally, a 3-D model is constructed of the object in the image, and that model is matched with stored 3-D models.

Acronym [Brooks *et al.*, 1979] is an object recognition system that follows Marr's approach quite closely. In the system, objects are modelled as generalized cylinders, and they are assumed to project as elongated ribbons in an image. Since the projections are elongated, the axis of the cylinder can be identified from the image, and an object-centered model of the object in the image can be constructed. That model is compared with stored 3-D models for recognition. However, since the objects modelled in *Acronym* are generalized cylinders and special properties of the projections of generalized cylinders are used, the system does handle general objects; it is difficult to construct a 3-D model of a general object from an image.

Other approaches to 3-D object-centered object recognition avoid this problem by not constructing a 3-D model of the object in the image. Rather, 2-D image features are matched with 3-D models. For example, Roberts [1965] finds edges in an image of simple polyhedra and matches them with model edges for recognition. Thompson and Mundy [1987] find pairs of dihedral vertices and match them with pairs of model vertices, computing a viewpoint and testing the match. Lowe [1986] takes a similar approach, except that he uses features which tend to be grouped together by people because their projections have invariant properties over wide ranges of viewpoints. Examples of such features include sets of parallel lines, collinear line segments, and line segments with close endpoints.

Object recognition using the aspect representation differs fundamentally from these approaches in that 2-D image features are matched with a representation of 2-D features of appearance, rather than with a 3-D model. This is possible because the asp represents appearance from all viewpoints rather than space occupied. Some effects of representing the appearance of an object are that the algorithm for object recognition is highly parallelizable and that features of occlusion such as T-junctions are helpful rather than harmful. For a survey of general 3-D object recognition see [Besl and Jain, 1985] or [Chin and Dyer, 1986].

The object recognition systems of Lowe [1986] and Thompson and Mundy [1987] can be classified as hypothesize-test systems, since they find a set of image features and hypothesize a match between that feature and a feature of the model, which then must be tested. A similar approach can be used in asp object recognition. However, it differs in that 2-D image features are matched with the asp representation of 2-D features rather than with 3-D model features. Matching 2-D features to 3-D features is more difficult since the 3-D representation of a feature may differ from the way it appears in an image. For example, an intersection point of several edges in the image may match a corner in the model, but image intersection point cannot just be matched with a corner with the same number of incident edges since some of the edges may not be visible in the image.

Another benefit of the asp is that since it represents the appearance of the object, it stores self-occlusion information, and this information can be used for object recognition. For example, in an image of an industrial part with holes, because of occlusion some edges will appear to intersect (i.e. intersect in the image) even though they do not intersect in the object. These intersecting lines are misleading for object recognition using object-centered representations since such representations don't have any corresponding pair of intersecting lines to look for. However, intersections in the image resulting from self-occlusion are explicitly represented in the asp, so features involving self-occlusion are more easily matched with the asp. In fact, representing occlusion accounts for the larger size of the asp; if an object has no self-occlusion it is convex, and the asp has the same size as object-centered representations.

An additional benefit of the asp is that comparing the image to the asp at a particular viewpoint is easier than comparing the image to a view of an object-centered representation, since in the latter case it is necessary to determine the appearance of the object by removing hidden lines. That is, the "test" part of "hypothesize-test" involves projection and hidden-line elimination, while in the asp hidden lines are not represented at all.

The aspect representation is reminiscent of multiview representations since both are viewer-centered and any appropriate cross section of the asp is a view of an object. Recognition in both cases involves matching 2-D features with a representation of 2-D features. However, there are some important differences. One difference is that the asp represents appearance continuously over all viewpoints, rather than from a discrete set of viewpoints. Furthermore, a representation of every topologically-distinct view of an object guarantees a topological match of the image with some stored view, but that requires a great deal of storage. And even if the image and a stored view match topologically, one cannot search for geometric features of appearance, such as lengths of lines, angles, and so on. With the asp one can search for geometric features of appearance since every appearance is represented.

Surprisingly, size is another benefit of the asp over multiview representations which represent all topologically-distinct views of an object. It may seem counter-intuitive that the asp should be smaller when it stores the appearance of the object from *all* viewpoints, but there are differences in efficiency and locality of information available. In the multiview representation, a whole image is necessary to represent one topological change in the appearance of the object. In the asp, just the change is represented. Also, the visual events bounding any viewpoint are not locally represented in the asp, as they are in the aspect graph. However, that information is not necessary for highly parallel algorithms, and additional storage is required to store it explicitly.

Thus, the size of the asp for a convex object with **n** faces is $O(\mathbf{n})$. However, there are $O(\mathbf{n}^2)$ topologically-distinct views of a convex object in the orthographic model, and representing every one may require as much as $O(\mathbf{n}^3)$ space (see Chapter 3). In the non-convex case, there are $O(\mathbf{n}^6)$ topologically-distinct views of an object in the worst case, and each view can have size $O(\mathbf{n}^2)$, so a multiview representation for a non-convex object can be as large as $O(\mathbf{n}^8)$. The size of the asp for an object is $O(\mathbf{n}^4)$ in the worst case.

4-5. Discussion

The difficulty of fast object recognition is that very few steps may be used in matching an image of an object with a stored representation. One approach to this problem is the use of viewer-centered representations for object recognition—representations of the way that an object will appear in an image under various viewing models. Fast object recognition is then accomplished through simple comparisons of various image properties. In this chapter we showed how to use the asp as a representation of appearance for fast object recognition, as an approach to the problem of fast object recognition.

The asp is similar to multiview representations in that it is viewer-centered and represents appearance. However, since the asp represents appearance from all viewpoints continuously, it enables algorithms to perform geometric rather than topological matches between the model and the image. The asp also has much smaller size than multiview representations that use all topologically-distinct views of a polyhedron.

Object recognition using the asp involves detecting distinctive features in the image and locating the corresponding features in the asp. For each hypothesized match, we find the viewpoints which make the appearance of the object feature match the image feature, and then compare the entire image with the asp cross-section to verify the match. The algorithm is stated in terms of parallel image feature detectors and associative memory, so that with sufficient processors recognition occurs "quickly," i.e. within a constant number of steps.

The asp algorithm for object recognition differs in some fundamental ways from methods based on object-centered models. Matching image features with model features is easier with the asp, since this matching is a comparison of 2-D features of appearance with a representation of appearance rather than a representation of the space that the object occupies. Another difference is that self-occlusion is represented explicitly in the asp so that image features generated by self-occlusion can be matched with asp features. Finally, since the asp represents appearance, we do not need to perform expensive runtime projection and hidden-line removal operations in order to compare model features with image features.

Marr [1978, 1982] has argued that object-centered representations are to be preferred over viewer-centered representations for object recognition. By *viewer-centered representations* he means multiple-view representations of objects. His argument is essentially that object-centered representations are more concise and therefore more desirable, and since humans can recognize objects from a wide variety of views and against a wide variety of backgrounds, they must use object-centered representations, demonstrating that they are sufficient.

However, the aspect representation is quite different from multiple-view representations and has significant advantages over them. First, the asp is not really any more "viewer-centered" than volumetric representations in his sense of the word—it doesn't depend on the location of the viewer; it represents appearance from *all* viewpoints. Furthermore, it appears that humans may have sufficient memory to speed recognition by storing several views of known objects, perhaps in addition to an object-centered representation. Finally, it is difficult to see how recognition with an object-centered model could occur as quickly as humans can recognize objects, while it appears that recognition with a viewer-centered model may be able to do so.

In this chapter we have considered the problem of recognizing a single object in an image. Since the asp represents appearance, self-occlusion is represented in the model. However, if several objects appear in a scene, objects may occlude one another. This sort of occlusion is not represented in the model and must be handled with techniques similar to those used in other approaches to object recognition. In fact, the local feature matching approach taken in this chapter has reasonable immunity to inter-object occlusion. Since feature are matched locally, an object may be recognized when only a part of it is visible; the visible parts cause a match and the invisible parts do not detract from the match.

In the model of visibility used in this chapter, edges of polyhedra are visible in the image as a result of low-level vision processes. This model is used in order to study the representation and use of shape in object recognition. This is perhaps a plausible model for polyhedra, since polyhedral edges can be detected through the use of intensity edges, depth edges, depth gradient edges, and so on. However, the data received from low-level sensors will contain data with noise and errors; one direction for future work is in dealing with such errors.

Another possible direction for future work is in the application of this type of algorithm to object recognition under different viewing models. Most likely, low-level vision will result in a number of maps of various sorts corresponding to different viewing models, such as edge maps, occluding-contour maps, color maps, and so on.

For example, a sensor system might be constructed that finds occluding edges in an image. Thus, a viewing model could be defined in which only occluding edges appear in images, and this type of algorithm could be used for recognition under such a model by constructing an analog of the asp under this model and using it for recognition.

Chapter 5: Animating Rotation

5-1. Introduction

The three-dimensional structure of an object is easier to perceive when the object is rotating or when the viewer moves to see the object from a range of viewpoints. For example, in CAD systems it is usually possible to see the object as it rotates. The 3-D shape of a human organ acquired from a CT-scan is more clearly perceived by displaying it as it rotates [Farrell *et al.*, 1985]. Displaying an object from a series of viewpoints (or "animating rotation") is also important to graphical simulations such as flight simulators.

In problems such as these, real-time display and hidden-line or hidden-surface removal for complicated polyhedra are desirable, but sometimes incompatible, goals. Flight simulators succeed at real-time display, but they generally do not perform true hidden-surface removal. Rather, they use domain-specific techniques such as displaying objects in a priority order so that objects in back will appear behind objects in front. They also use relatively simple polygonal models objects, relatively slow frame-rates, and expensive hardware [Yan, 1985].

In this chapter, we present an algorithm for animating the rotation of a polyhedral object or scene by displaying a series of line-drawings of the scene from viewpoints along a path in viewpoint space. The first image of such an object is shown in Figure 5-1.

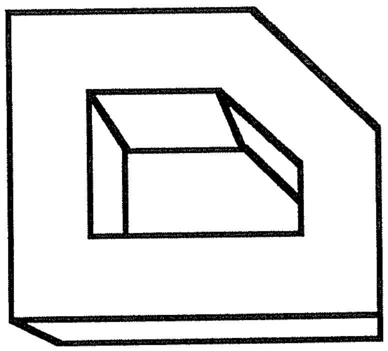


Figure 5-1. The first image of an object that is to be rotated.

Subsequent images are also line drawings, formed from closely-spaced viewpoints along the path.

The naive approach to animating rotation is to treat frames independently. For each frame, hidden lines or surfaces are removed and the frame is displayed. The real-time goal is to do this at a real-time rate, such as 30 frames per second. However, there may be very little difference between two frames from nearby viewpoints. In many cases, successive frames will be identical except for the linear change in the location of the edges caused by the change in viewpoint. In such cases, hidden-line removal is superfluous since the result will be the same as the previous frame except for the changes in vertex positions.

In this chapter we present an algorithm for animating rotation that takes advantage of this frame-to-frame coherence by computing the initial appearance of the scene in the first frame and the differences between successive frames. That is, given a path in viewpoint space and a set of polyhedra or a polyhedral scene, we present an efficient method for displaying images of the scene from a series of viewpoints along the path. The images consist of the line segments of the scene that are visible from each viewpoint. The algorithm can also be used for display of shaded faces. We use orthographic projection and assume that the path of viewpoints forms a great circle in viewpoint space. Extensions to perspective projection and other paths of viewpoints are also discussed.

The algorithm for animating rotation has two phases, a preprocessing phase and an on-line phase. In the preprocessing phase, the appearance of the polyhedron from every point along the path of viewpoints is computed and represented as the asp defined for a 1-D viewpoint space (the path of viewpoints). All of the events or changes in the topology of the image that occur along the path of viewpoints are represented as vertices of the asp. A list of vertices sorted by viewpoint along the path is also kept, in effect a 1-D viewpoint space partition (see Figure 5-2).

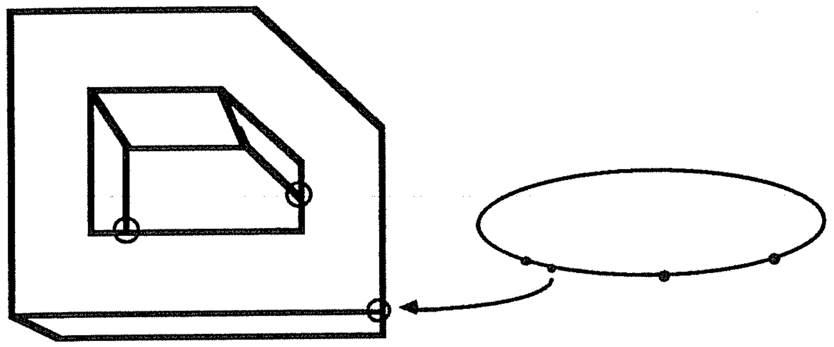


Figure 5-2. Visual events along the path of viewpoints are computed and stored.

The final step of the preprocessing phase is to determine which of the asp features are visible from the initial viewpoint. This is done by testing each face with the initial viewpoint, and keeping a list of pointers to the visible faces.

The second phase is the on-line display of frames or images from closely-space viewpoints. Displaying subsequent frames involves comparing the new viewpoint with the list of events. If the viewpoint does not pass any events, a linearly-shifted version of previous image is displayed. If the viewpoint does pass an event, the list of visible faces is updated according to the event passed, and the updated image is displayed from the new viewpoint (see Figure 5-3).

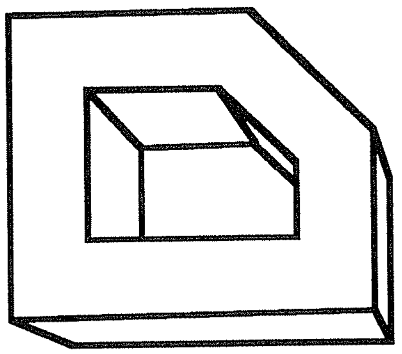


Figure 5-3. When the viewpoint passes events, the image is updated and displayed from the new viewpoint.

In Figure 5-3, the viewpoint has passed an event as compared to the viewpoint of Figure 5-2, and two new edges have appeared.

Section 5-2 of this chapter contains a discussion of how to use the asp to find images of the scene from nearby viewpoints. Section 5-3 shows how to use the asp to animate rotation along a 1-D path of viewpoints. Section 5-4 discusses the results achieved, and Section 5-5 discusses extending the algorithm to other viewpoint paths and to perspective projection. Section 5-6 contains a discussion of related work. Section 5-7 contains a summary and concluding remarks.

5-2. The Asp and Changing Appearance

Every cross-section of the asp for a polyhedral scene at a fixed viewpoint is the appearance of that scene with hidden lines and surfaces removed. In that sense the asp contains all images of the scene in an explicit form. Finding an image of the scene from a given viewpoint is reduced to finding a cross-section of the asp. The difficulty is in finding the parts of the asp that are visible from a given viewpoint: the asp is a large data structure, and without further information finding the cross-section is a linear-time operation.

However, in animating rotation the problem is simplified to some extent. Finding the first view still requires linear time in the size of the asp, but subsequent views are "nearby" in the sense that the viewpoint at which the cross-section is taken is close to the previous viewpoint. Thus, not much has changed in the cross-section of the asp. The only problem is in efficiently finding the parts of the image that have changed because of the change in viewpoint. That is, the problem is in knowing which events are near any point in viewpoint space.

The viewpoint space partition or VSP (see Chapter 3) is a partition of viewpoint space into regions of constant aspect, and the boundaries of regions correspond to visual events. Thus, the VSP contains the additional information needed. With pointers from the boundaries of the VSP to the faces of the asp involved in the corresponding event, the events passed by a moving viewpoint can be found efficiently.

Using this data structure, it is possible to animate rotation efficiently. The first view is found by testing every asp facet to determine whether it is visible using the algorithm given in Chapter 3. The particular region of the VSP in which the viewpoint lies is also found by linear search. Then to display subsequent views, we determine whether the path of viewpoints has passed through any boundaries of the VSP since the last view by finding any intersections between the viewpoint path since the last view and the boundary of the current region. If the path of viewpoints passes into another region, the list of visible faces is updated according to the visual event passed. The path of the viewpoint through the VSP is traced until the new viewpoint is reached, updating the set of visible faces as necessary. The next view is then displayed by displaying the set of visible object edges, represented by asp facets.

This algorithm animates rotation along any path in viewpoint space. It involves constructing the VSP for the scene, so for a scene of **n** faces the preprocessing phase requires $\Omega(\mathbf{n}^2 \log \mathbf{n})$ to $O(\mathbf{n}^6 \log \mathbf{n})$ time, depending on the visual complexity of the object. When the viewpoints are closely spaced so that not many events are crossed per change in viewpoint, displaying subsequent frames requires linear time in the output size, i.e. the number of edges or regions displayed.

5-3. Animating Rotation along a 1-D Path of Viewpoints

Because of the high time-complexity of the preprocessing phase, the algorithm sketched above for animating rotation using the asp and the VSP is practical only for problems that are small or visually not very complex, or for which a very large number of views will be required. However, it is possible to improve the time required for the preprocessing phase and reduce the storage requirements when the path of viewpoints along which views will be generated is known in advance.

The asp represents appearance as a function of viewpoint, where viewpoint has two degrees of freedom under the orthographic model. Similarly, the VSP is a partition of a 2-D viewpoint space. However, the path of viewpoints is 1-D, and the algorithm really only requires a representation of appearance and the VSP along this 1-D path. In this section we will assume that the desired path of viewpoints is a great circle in viewpoint space. In fact, we will assume that the rotation is about the y-axis (i.e., a rotation of θ with $\phi = 0°$). Other rotations can be handled by rotating the scene so that the desired path of viewpoints is strictly a θ rotation, with $\phi = 0°$. The scene is rotated back before display. We will show how to construct the asp and the VSP for this 1-D viewpoint space.

5-3.1. The Asp under a 1-D Orthographic Model

The asp for a polyhedral scene constructed under the 1-D orthographic model is a set of cells of a 3-D aspect space consisting of an image plane for every viewpoint along the path. Since the path consists of a θ rotation with $\phi = 0°$, the asp is like the asp of Chapter 2 with ϕ constrained to be 0°. For example, with $\phi = 0°$ the asp for a point (x_0,y_0,z_0) becomes

$$u = x_0 \cos \theta - z_0 \sin \theta \qquad (5\text{-}1)$$

$$v = y_0 \qquad (5\text{-}2)$$

Figure 5-4 shows the asp for a triangle under the 1-D orthographic model. Since the asp is a 3-D cell rather than 4-D, Figure 5-4 shows the whole asp and not a cross-section.

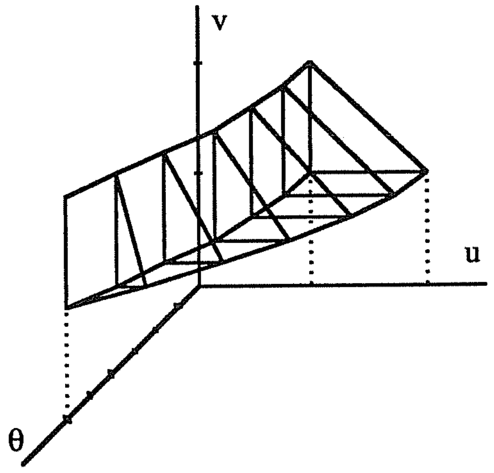


Figure 5-4. The asp for a triangle under the 1-D orthographic model.

In the 1-D orthographic model, viewpoint space is 1-D, and the image plane is still 2-D, so aspect space is a 3-D space rather than 4-D. The asp features corresponding to features of appearance are of lower dimension. A 3-D cell of aspect space corresponds to a polygon in the image, and the edges bounding polygons are represented by 2-D faces bounding the cells. Vertices and T-junctions are 0-D visual events visible from a 1-D region in viewpoint space, so they are represented by 1-D asp faces, i.e. edges. Three object edges appear to intersect from single viewpoints, so those visual events are represented by vertices in the asp.

Under the 2-D orthographic model, three edges appear to intersect in a single point from a 1-D path of viewpoints (see Figure 5-5). This visual event is the general kind of 1-face of the asp under that model.

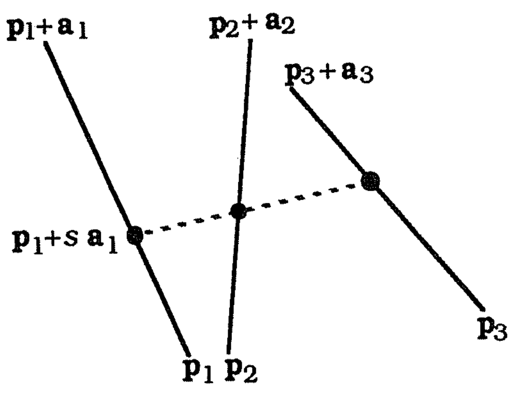


Figure 5-5. The viewpoints from which three object edges appear to intersect in a single image point.

The viewpoints from which these edges appear to intersect are given by Eq. (5-3).

$$\mathbf{V}' = ((\mathbf{p}_1 + s\,\mathbf{a}_1 - \mathbf{p}_2) \times \mathbf{a}_2) \times ((\mathbf{p}_1 + s\,\mathbf{a}_1 - \mathbf{p}_3) \times \mathbf{a}_3) \tag{5-3}$$

In the 1-D orthographic model, the viewing direction is constrained to lie in the xz-plane. The form of Eq. (5-3) is

$$\mathbf{V} = \mathbf{a}\,s^2 + \mathbf{b}\,s + \mathbf{c} \tag{5-4}$$

Since $V_y = 0$, the viewpoints at which the three lines appear to intersect can be found by solving Eq. (5-5) for *s*:

$$V_y = a_y\,s^2 + b_y\,s + c_y = 0 \tag{5-5}$$

Thus there are at most two viewpoints (and their polar opposites) from which three lines appear to intersect in a single point. These are the general sort of asp vertices.

The asp under this model is constructed in the same way as the asp under the 2-D orthographic model. That is, in order to construct the asp for a face **f** of a scene, the asp for each face (by itself) is first constructed. The form of the asp for a face is displayed graphically in Figure 5-4. Then to construct the asp for **f** as occluded by the other faces, the asps for the other faces are subtracted from the asp for **f**. The difference in this case is that the cells are 3-D in a 3-D space, rather than 4-D in a 4-D space. This is a simplification; in fact, finding the intersection of polyhedra or 3-polytopes is a necessary step or subproblem of finding the intersection of 4-polytopes. Finding the intersection of 3-cells of size **m** and **n** still requires O(**mn**) time.

Since there is a constant number of asp vertices for every set of 3 object edges, the number of vertices is bounded by O($\mathbf{n}^3$) where **n** is the number of polygons in the polyhedral scene. These are the vertices of a 3-manifold in 3-space, so the number of edges, faces, and cells is also bounded by O($\mathbf{n}^3$) by Euler's formula. Letting **q** be the size of the asp for a face **f** and noting that the asp for a face of size **r** by itself is O(**r**), the time to subtract the asp for a face from the asp for **f** is O(**qr**). Thus, the time to construct the asp for the face is $O(\mathbf{q}\ (\Sigma \mathbf{r})) = O(\mathbf{qn})$. The time for the construction of the asp for the whole scene is $O(\Sigma \mathbf{qn}) = O(\mathbf{n}^4)$. The convex case requires less time: there are no faces in front of any other face, so there is no subtraction of cells to be done. However, if the polyhedron is not known in advance to be convex, the naive algorithm for finding all of the faces in front of each face takes time $\Theta(\mathbf{n}^2)$.

5-3.2. The VSP under a 1-D Orthographic Model

Under the 2-D orthographic model, the VSP is a partition of the 2-D viewpoint space into regions of constant aspect. Under the 1-D orthographic model, it is a partition of the 1-D space into regions of constant aspect. Thus the boundaries are viewpoints at which events occur, separating the path of viewpoints into regions of constant aspect.

In the 1-D orthographic case, the events are viewpoints of the form shown in Figure 5-5, and they are represented as vertices in the asp. Thus, to find the VSP it is sufficient to project all of the vertices of the asp into viewpoint space. In other words, it is sufficient to find all of the viewpoints as given by Eq. (5-5) for all of the asp vertices. We will represent the VSP in this case as a sorted list of viewpoints. With each event viewpoint we store a pointer to the asp vertex at which the event occurs.

5-3.3. Animating Rotation

The preprocessing phase of animating rotation involves constructing the asp for a polyhedral scene under the 1-D orthographic model, finding the projections of the asp vertices into the 1-D viewpoint space, and sorting them. Pointers from the event viewpoints to the asp event are also kept. Finally, the set of visible edges in the initial image is found by a linear search through the asp 2-faces using the visibility testing algorithm of Chapter 2. A list of pointers to the visible faces is kept.

The on-line phase of animating rotation involves keeping track of the set of visible asp 2-faces (corresponding to image edges) as the viewpoint crosses events and displaying the set of visible edges. As each event is passed, the asp 2-faces that become visible are added to the list of visible faces, and the faces that become invisible are removed from the list. The particular asp 2-faces that appear or disappear depend on the particular arrangement of faces around the image point at which the event occurs, and the update can be handled as a set of special cases for the different events listed in Section 3-5.1.1.

This process is about as fast as displaying a wire-frame object if the number of events is no more than the order of the number of frames to be displayed. In fact, there are (usually) fewer edges to display, since some are not visible. Updating the list of visible faces is generally very fast; it takes a significant amount of time only when a very large number of edges appears or disappears at once. The algorithm takes full advantage of hardware that displays vectors quickly, and it can also make use of 3-D rotation hardware for display.

The time to update the data structure for an event is proportional to the number of edges that appear or disappear in the image. Usually, the number of edges that appear or disappear will be a small constant, say 2-3. The time to display an additional image is linear in the output size, i.e. the number of edges to be displayed.

The preprocessing time required is essentially the time to construct the asp for the scene and to sort the viewpoints at which events occur. Sorting the events takes less time than constructing the asp. Constructing the asp takes time $\Omega(\mathbf{n}^2)$ to $O(\mathbf{n}^4)$, depending on the number of visual events in the scene. A large class of objects (probably most common objects) have $O(\mathbf{n}^2)$ visual events, and the asp in this case can be constructed in $\Omega(\mathbf{n}^2)$ to $O(\mathbf{n}^3)$ time.

This algorithm requires storage for every visual event along the path of viewpoints. Thus, the amount of storage required for events will be between $\Omega(\mathbf{n})$ and $O(\mathbf{n}^3)$. The actual amount required varies according to the visual complexity of the scene. For convex scenes storage is not a problem until the number of faces reaches about 10^6; for the worst case of visual complexity storage becomes the limiting factor when the number of faces is in the hundreds. In practice, objects have a visual complexity much closer to the convex case. Most common objects have $O(\mathbf{n}^2)$ events, where the constant involved is less than one, so that a practical upper bound for which this algorithm is useful for such objects is in the thousands of faces. These bounds can be increased in cases where many views are required and more storage is available. Examples of such problems are flight simulators and video games. Some of the storage can be slower than main memory because the memory in active use at one time is only about as much as the memory required for one image and a small part of the VSP. Note that the total storage requirement for a series of frames is much less than that required to store multiple frames.

The on-line display portion of this algorithm is completely parallelizable; with a processor for each visible edge, frames can be displayed in constant time plus the time to update the list of visible edges according to any events passed. With fewer processors, speedup is optimal for the number of processors used. The asp construction portion of the algorithm can also be implemented in parallel: the asp for each face of the scene is constructed independently, and a processor can be assigned to the construction of the asp for each face in parallel.

5-4. Results

The on-line portion of this algorithm has been implemented. In order to determine the time required to animate rotation with fast 2-D vector-drawing hardware, we measured the time to compute the vectors but not to draw them. On a Sun 4/110 workstation, the vectors were computed at a rate such that scenes with about 2000 faces of moderate visual complexity could be displayed at 30 frames per second.

This algorithm is one to two orders of magnitude faster than existing algorithms for animating rotation for a class of polyhedra. The fastest results reported in related work are a frame rate of 1-2 seconds per frame for a scene with 1000-2000 faces (see Section 5-6 for a discussion of related work). The main reason for the much greater speed is that the algorithm is vector-based rather than raster.

5-5. Extensions

The algorithm can be extended to work for paths of viewpoints that are not great circles on the sphere. Still using the orthographic model, it is possible to find the asp along any path in viewpoint space such that the intersection of that path and the faces of the asp can be found in closed form. If aspect surfaces and their intersections can be represented, the asp can be constructed.

The algorithm can also be used under a perspective model. In this case it is necessary to be able to find the intersection of the path of viewpoints and the surfaces in aspect space under the 3-D perspective model. This would be useful, for example, in showing an animation in which the viewpoint moves through a workspace, such as a model of a house.

It would be much more difficult to extend the algorithm to work for scenes in which objects are moving. That would require the use of a higher-dimensional parametric space (one to three degrees of freedom in viewpoint, two degrees of freedom in the image, and a degree of freedom in time). Worse, depending on the motions of the objects, it may not be possible to solve for the intersections of the surfaces in closed form.

5-6. Related Work

We are concerned in this chapter with the efficient computation of images given a static scene and a path of viewpoints. This is distinct from the more common problem termed "animation" in computer graphics. This sort of animation is the problem of using computers to assist in "Walt Disney-style" animation. That is, it is the problem of computing the location of moving objects or people in successive frames to make motion look natural, an area on which much work has been done (see, for example, the bibliography by Magnenat-Thalmann and Thalmann [1985] or the text by Fox and Waite [1984]). In our problem, the scene is static and the viewpoints are known; we are interested in computing the images quickly.

In computing successive frames of an animation sequence, the standard technique is to compute each frame independently. Animated displays of moderate complexity are generally computed in advance and displayed in real time. Denber and Turner describe a method of compressing the data in an animated sequence and increasing the speed of their replay [Denber and Turner, 1986]. The technique involves storing and displaying only the difference between successive images. The technique is raster-based, and it does not involve a faster method for computing the successive images. Glassner describes a method for faster ray-tracing of a sequence of images in an animation [Glassner, 1988]. The technique uses a space subdivision algorithm for decreasing the time required for ray tracing. The novel part of the method is that it uses the subdivision techniques in 4-D *spacetime* rather than 3-D space, achieving around a 50% decrease in ray-tracing runtime for the examples given.

Flight simulators achieve real-time display of a static scene from a moving viewpoint, but existing systems do not usually perform general hidden-line removal. Rather, they assign scene objects an empirically-determined priority and display objects in the scene according to their priority, so that for example the ground is drawn before any objects on the ground [Yan, 1985]. For the kinds of objects and scenes involved, it is usually a sufficient technique, but it is not hidden-surface removal. In addition, professional flight simulators use expensive equipment and model relatively simple objects in the world in order to display the images in real time.

Zyda *et al.* [1988] describe a "low-cost" flight simulator, i.e. hardware cost of under $100,000. They state that this is at least an order of magnitude less expensive than the cost of existing systems. It uses a pipeline of 12 geometry engines to perform rotation, translation, scaling, clipping, and orthographic and perspective viewing. This system displays a scene consisting of around 1500-2000 polygons at 3-4 frames per second. However, the system "does not currently have real-time hidden-surface-elimination hardware except in rudimentary form." It accomplishes hidden surface elimination by drawing the scene in farthest-to-closest order using the priority assignment technique.

Hubschman and Zucker introduce the idea of using frame-to-frame coherence to decrease the time required for hidden-surface removal [Hubschman and Zucker, 1981]. They work in a world with one or two stationary convex polyhedra, and they find a number of frame-to-frame coherence constraints. The result is a partition of the scene such that "the movement of the viewing position across a partition boundary results in an occlusion relationship becoming active or inactive." The scene is updated when one of these "change boundaries" is crossed. This approach is similar in spirit to our initial VSP approach of Section 5-2, except that it is developed for the restricted world of two convex polyhedra. As a result, the storage requirements are $\Omega(\mathbf{n}^3)$ in the worst case in this simplified world, perhaps higher. A generalization of this technique to non-convex polyhedra would result in worst-case storage requirements of $\Theta(\mathbf{n}^9)$ for a scene with $\mathbf{n}$ faces.

Shelley and Greenberg introduce the idea of using frame-to-frame coherence for animation with a viewpoint moving along a path of viewpoints in viewpoint space [Shelley and Greenberg, 1982]. They use a number of ad hoc culling and sorting rules to reduce the work in hidden-surface removal. For example, they find the box bounding the path of viewpoints. Then any faces that point away from all viewpoints in the box can be removed from consideration for all viewpoints along the path.

Fuchs *et al.* address the problem of displaying a set of polygons from an arbitrary viewpoint in near-real time [Fuchs *et al.*, 1983], with an approach that involves constructing the BSP-tree (Binary Space Partition tree) [Fuchs, *et al.*, 1980] for a scene in an off-line preprocessing stage. Then the display of a frame from some viewpoint with hidden surfaces removed involves traversing the tree to get a list of faces in an approximation of back-to-front order. The faces are all drawn on the screen, and the result is an image with hidden surfaces removed. This approach is much better than the naive approach in that the work involved in displaying an image is a tree-traversal and display of all of the faces of the polyhedron.

The BSP-tree approach has some drawbacks, however. The number of faces in the tree is larger than the number of faces in the scene since some faces have to be split to construct the tree. According to Fuchs *et al.* the result is less than a factor of two increase for the example scenes shown; for more complex scenes the increase may be greater. The theoretical worst case is an $\Omega(\mathbf{n}^2)$ increase. Also, the approach works only for hidden-surface removal and shaded display of the polyhedron. Shaded display is often desirable, but it is slower than displaying line segments with hidden lines removed. As a result, the algorithm achieves frame-times of 1-2 seconds per frame for up to 1000 initial polygons on an Ikonas RDS3000 raster graphics system, based on a programmable AM29000-based processor. Thus, it does not achieve real-time rates for scenes of complexity on the order of 1000 polygons.

5-7. Concluding Remarks

This chapter presented an algorithm for efficiently animating rotation that takes advantage of the frame-to-frame coherence inherent in an animated sequence. Rather than treating each frame as independent and computing the appearance of the scene from each viewpoint independently, the algorithm computes the appearance for all viewpoints by computing the asp for the scene. In order to reduce storage requirements, the asp is constructed for a 1-D path of viewpoints, and images of the scene are displayed from viewpoints along that path.

The only previous algorithm for efficiently animating rotation of general polyhedra with hidden lines or surfaces removed is that of Fuchs *et al.* [1983]. After some pre-computation time, it is efficient at computing the faces of the scene in an order that approximates back-to-front. This is sufficient for hidden surface removal by drawing all of the (shaded) faces from back to front, but not for hidden line removal, and hidden line removal is required for vector visible edge display, which can be done much more quickly than raster shaded face display. Another inefficiency with the algorithm of Fuchs *et al.* is that all of the faces are drawn, not just the visible one. In fact, since some of the faces are split, twice as many faces or more may be drawn as are in the scene. The results reported by Fuchs *et al.* are a frame rate of 1-2 seconds per frame for a scene with 1000-2000 faces.

Our asp algorithm for animating rotation is many times faster than that of Fuchs *et al.* for a class of polyhedra. It is also much more efficient in its use of space than storing multiple views of a scene. The disadvantages of the algorithm as compared to the BSP algorithm are that (1) preprocessing time is required for a particular path of viewpoints rather than for views in any order, and (2) storage requirements are greater.

The algorithm presented here is efficient only for a certain class of scenes. When the number of faces in the scene becomes large and the scene is visually complex, the number of visual events can become too large to store. In the convex case, the number of visual events is $O(\mathbf{n})$ where $\mathbf{n}$ is the number of faces in the scene, but in the worst case the number of visual events is $O(\mathbf{n}^3)$.

Chapter 6: Related Work

6-1. Introduction

In this chapter we present an overview of work related to object representation and the aspect graph. Work related to the applications of Chapters 4 and 5, object recognition and animating rotation, has been presented in the chapters themselves. In addition, references to the most pertinent work related to the aspect representation and the aspect graph have been included in the corresponding chapters. This chapter is intended to provide a more thorough overview of related work. Section 6-2 contains a bibliography of object representation in general, and Section 6-3 discusses object-representation techniques with viewer-centered ideas. Section 6-4 discusses work on *visibility*, the determination of the parts of a scene visible from a point or a line. This work is related to these ideas since it involves computing and representing appearance from a viewpoint or viewpoints. Section 6-5 discusses work related to the aspect graph.

6-2. Object Representation

In this section we describe current techniques commonly used for representing solid objects. Requicha presents a survey of object representation techniques [Requicha, 1980] and Badler and Bajcsy survey object representations for computer vision [Badler and Bajcsy, 1978].

Current techniques for representing solid objects can be separated into three major categories: boundary representations, swept-volume representations, and volumetric representations. Boundary representations represent a polyhedral object by the faces that bound it. They represent faces by the bounding line segments and line segments by the endpoints. The representation is thus a data structure consisting of many points (the vertices of the polyhedron) and adjacency and face inclusion information. If the adjacency information stored in the data structure specifies only the endpoints of each line segment, then the representation is called a *wire-frame* representation since the result of displaying these edges looks like a wire-frame model of the object. It is in general not possible to construct an image of an object with hidden lines removed from this sort of representation, since wire frames are ambiguous, that is, a wire-frame model may correspond to several solid objects [Markowsky and Wesley, 1980].

If the adjacency information also specifies the edges that bound each face, then the representation is no longer ambiguous, and it is possible to draw an image of the object with hidden lines removed. A popular and commonly-used representation for polyhedra is the *winged-edge polyhedron* representation [Baumgart, 1972], which stores all the adjacency information in space proportional to the number of edges of the polyhedron and for which all adjacency relations can be determined in time proportional to the number of items in the relation.

Boundary representations can also be used to represent non-polyhedral objects by representing the surfaces with various types of surface patches. For example, Potmesil [1979] uses bicubic patches, and Levin [1979] and Dane and Bajcsy [1981] use quadric-surface patches. Another common method for representing curved surface patches is with B-splines [Coons, 1974].

Swept-volume representations represent a solid by representing an infinite union of cross sections by means of a cross sectional function and an axis or spine of the object. The cross section may be constant along the whole axis, it may be allowed to change linearly from one end of the axis to the other, or it may be stored as a number of samples or some other function. This idea was first introduced as the "generalized cone" by Binford [1971]; current examples of the use of generalized cones and cylinders include work by Marr and Nishihara [1976], Agin [1977], Soroka and Bajcsy [1978], Brooks *et al.* [1979], and Shafer and Kanade [1983].

One type of volumetric representation is the spatial occupancy representation. In this type of representation, the volume of space that an object occupies is represented by cutting up space up into a number of cubes or other cells and representing the particular cells of space that the object fills. In the most straightforward approach, a 3-dimensional array of bits represents a number of cubes filling a volume of space. Then to represent an object, one sets the bits for the cells of space that it fills to 1 and the rest to 0 [March and Steadman, 1974; Srihari, 1981]. A more space-efficient approach is the octree [Jackins and Tanimoto, 1980]. In the octree representation for an object, space is divided into eight octants. If the object fills any of these octants or any is empty, then that is represented. If the object partially fills an octant, then that octant is divided into eight sub-octants, and the process is repeated in a hierarchical manner until the desired precision is achieved. Space savings over the straightforward approach occur because it is only necessary to store a large number of small cubes near the surface of an object rather than throughout all of space.

Another sort of volumetric representation is constructive solid geometry [Requicha, 1978]. This technique represents the volume of space that an object fills as the union, intersection, and difference of a fixed number of primitive solid types, such as cylinders, blocks, spheres, and cones. The object is represented as a tree with nodes representing Boolean operations and leaves representing primitive types. This sort of representation is often used in geometric modeling systems [Requicha and Voelcker, 1982; 1983].

6-3. Object Representation Techniques with Viewer-centered Ideas

All of the major object representation techniques presented above are object-centered in the sense that they represent the volume of Euclidean space that the object occupies, either directly or indirectly. The asp, on the other hand, represents appearance. In this section we discuss work of a more viewer-centered flavor, that is, work concerned with representing the way an object appears from various viewpoints or making it easier to calculate the appearance of an object.

Various techniques have been proposed for representing objects in a tree-structure of some sort in order to simplify calculations of surface intersections or hidden-line removal. Clark [1976] proposes using a hierarchy of bounding volumes, each of which contains the lower levels. The bottom level of the hierarchy contains a different sort of representation of the object. Rubin and Whitted [1980] use a tree of parallelepipeds, with each level containing the levels below it. Fuchs *et al.* generate a "binary space partition tree" which represents faces or parts of faces in a tree in order to speed detection of intersection with a ray [Fuchs *et al.*, 1980]. This tree also allows rapid retrieval of faces in an approximation of back-to-front order for z-buffer hidden surface removal [Fuchs *et al.*, 1983]. Ponce [1985] creates a hierarchical *prism tree* which also speeds intersection detection. While these techniques do not represent appearance directly, their goal is to facilitate computing different views of an object or scene.

The Extended Gaussian Image (EGI) is a representation that has been introduced for 3-D object recognition [Ikeuchi, 1981; Horn, 1984]. A mapping is defined between the orientation of points on the object and points on the Gaussian sphere. The EGI is a representation of the surface orientation density for an object on the Gaussian sphere. The EGI can be approximated with a surface orientation histogram for an object. The EGI does not represent appearance, but it does represent a property of the appearance of an object from all viewpoints.

In another approach to fast 3-D object recognition, "multiview" or "characteristic view" object representations have been introduced. The idea is that in order to represent a three-dimensional object one can store a number of silhouettes or two-dimensional images with hidden lines removed. This idea of "characteristic views" was introduced by Lavin [1974], and Perkins [1978] and Holland *et al.* [1979] describe early systems making use of multiple views. Lieberman [1979] computes silhouettes of objects from stable orientations, assuming that there are only a few. Wallace *et al.* [1981] compute properties of the silhouettes of aircraft from a number of fixed viewpoints. Fekete and Davis [1984] introduce the idea of *property spheres*.

The property sphere represents the silhouette of an object from 320 discrete viewpoints spaced around the sphere. Kim *et al.* [1985] outline a system for recognizing moving objects with a moving camera using multiple views of an object. Work has also been done on multiview representations that store a view for every vertex of the aspect graph; this work is discussed in Section 6-5.

The multiview representation idea is viewer-centered in the sense that it involves representing the way an object appears to a viewer from a number of different viewpoints. We know of no representation other that the asp that represents the appearance of an object as a function of viewpoint or from all viewpoints.

6-4. Visibility

A visibility problem is a problem that involves determining the parts of a scene that are visible from a given viewpoint or viewpoints. Problems of visibility also have a viewer-centered flavor, since they must compute and represent appearance. Hidden-line and hidden-surface removal are visibility problems, essentially the problem of computing visibility from a given viewpoint. Much work has been done on this problem in the computer graphics literature. Sutherland *et al.* characterize ten hidden-surface algorithms [Sutherland, 1974]; see also the computer graphics text by Rogers [1985]. A recent algorithm for hidden-line removal has $O(\mathbf{n}^2)$ worst-case time complexity [Dévai, 1986].

El Gindy and Avis [1981] and Lee [1983] compute the "visibility polygon" from a point in the plane, that is, the list of edges visible from the point. Avis and Toussaint [1981] compute visibility from many viewpoints simultaneously when they discuss the visibility of a polygon in the plane from viewpoints along an edge in the plane. However, their algorithm computes the parts of a polygon that are visible from *some* point along a line in the plane, rather than visibility as a function of viewpoint along the line. Consequently, their work does not really compute visibility from a continuous region of viewpoints. Other work on the visibility polygon from an edge includes work by El Gindy [1984], Chazelle and Guibas [1985], Guibas *et al.* [1986], and Suri and O'Rourke [1986].

Canny [1984] computes the region of a plane of viewpoints from which a polygon in R^3 is visible. He discusses computing the "umbra" and "penumbra" of a polygonal face of a polyhedron relative to a polygonal light source. The umbra is the region of partial visibility and the penumbra is the region of total occlusion of the face. He gives an algorithm for constructing the "shadow" of a polygon relative to another polygon on the plane of viewpoints. However, his method does not construct the shadow caused by more than two polygons because he does not consider edge-edge-edge events in that paper.

In other work on visibility, Lumia *et al.* [1985] present an algorithm for answering queries about whether a feature point on an object is visible from various viewpoints. The algorithm pre-computes the regions of viewpoints at which each face obstructs the view of the feature to enable a faster answer to the queries. Hubschman and Zucker [1981] are interested in decreasing the time required for hidden-surface removal with a moving viewpoint and a static scene consisting of two convex polyhedra, and they compute boundaries in space at which the appearance of the scene changes. These two papers are related to the VSP and representing appearance as a function of viewpoint since they partition viewpoint space into regions in which some property of appearance is constant.

In related work, Thorpe and Shafer [1983] analyze the topological constraints on the change in appearance of an object as the viewpoint moves. Scott [1984] presents a system which tries to understand the topology of the projection of an object with changing viewpoint.

6-5. The Aspect Graph

Koenderink and van Doorn [1976] laid the groundwork for the study of the topologically-distinct views of an object when they introduced the idea of the "singularities of the visual mapping" for a smooth body. The singularities of the visual mapping are points of the image that map to points in the world at local minima and maxima of distance and points on surfaces tangent to the viewing direction. They catalog the ways that the topological structure of the singularities can change and show that for most vantage points the structure of the singularities does not change with small changes in viewpoint. They call a change in the topological structure an "event," and they catalog the different kinds of events that occur for smooth bodies.

In a later paper [1979] Koenderink and van Doorn define an "aspect" as the structure of the singularities of a stable view of a smooth object, and they define the "visual potential" graph for such an object, which we call the aspect graph. However, their domain of objects is different from ours since they assume that objects are smooth and we assume that objects are polyhedral. In addition, their model of visibility is different from ours. We assume that the edges of a polyhedron are visible, while they are concerned with the "singularities of the visual mapping," which are image points on occluding edges or contours and image points corresponding to object points on surfaces that are tangent to the line of sight. However, for polyhedra their definition of aspect and ours are similar, and an example of the aspect graph that they give for a tetrahedron was the inspiration of our definition.

In Chapter 3 we discussed the VSP, which Koenderink and van Doorn do not define or discuss in detail. They do mention something similar, however, when they state that "a general decomposition of [viewpoint space] into cells that provide a stable global aspect is by no means trivial to carry out. To our knowledge this problem has not yet been solved by geometers." [1976, p. 57]. The cells of stable global aspect are the cells of what we call the VSP, except for the differences in the types of objects we allow and in the kind of events we recognize.

Some work has been published on constructing the aspect graph. Werman, Baugher, Gualtieri [1986] show how to construct an aspect graph for a convex polygon with viewpoint constrained to lie in the plane containing the polygon, using perspective projection. They also present some properties of the aspect graph in that case. McKenna and Seidel [1985] construct minimal and maximal shadows of convex polyhedra in the plane. In doing so, they project the great circles corresponding to the faces onto a plane and construct the resulting arrangement of the lines in the plane using the $O(\mathbf{n}^2)$ algorithm of Edelsbrunner *et al.* [1986] or Chazelle *et al.* [1985]. The result is similar to the VSP for convex polyhedra under orthographic projection. Kender and Freudenstein [1986] discuss the meanings of terms such as "degenerate view," "characteristic view," "visual event," and "general viewing position."

The algorithm of Chapter 3 for constructing the aspect graph has been presented earlier for the convex and general cases under orthographic projection [Plantinga and Dyer, 1986]. It is the first algorithm for constructing the aspect graph in the general case. The definition given there of aspect was slightly different that the definition given in Chapter 3, but the algorithm works under both definitions. Stewman and Bowyer [1987] present an algorithm for constructing the aspect graph for convex objects under perspective projection. No analysis of time requirements are given, but the algorithm appears to require at least $\Omega(\mathbf{n}^4)$ time. Thus it is less efficient than the algorithm of Chapter 3, which is optimal, requiring $\Theta(\mathbf{n}^3)$ time in the worst case.

Watts [1987] discusses the "principal views" of a polyhedron, which are similar to aspects except that none are allowed to be identical. (Two aspects can have the same appearance if the corresponding regions of viewpoint space are separate.) An algorithm is presented for constructing the principal views of a convex polyhedron under perspective projection that runs in $O(\mathbf{n}^4)$ time. Recently, Gigus and Malik have presented an algorithm for constructing the aspect graph for non-convex polyhedra under orthographic projection [Gigus and Malik, 1988]. The definition of aspect used there is the same as the definition given in Chapter 3 and slightly different from that of [Plantinga and Dyer, 1986]. Their algorithm has the same worst-case runtime as the algorithm given in Chapter 3. However, our algorithm has better average-case run-time for simpler objects; its runtime for convex polyhedra is $O(\mathbf{n}^2 \log \mathbf{n})$, and the runtime of the algorithm of Gigus and Malik is $\Omega(\mathbf{n}^3)$.

The aspect graph has been used as way of selecting views for multiview object recognition. When a view is stored for every aspect of the aspect graph, some view is guaranteed to match the image topologically, and the number of views stored is not larger than necessary for topological matching. Using these ideas, Chakravarty and Freeman [1982] assume that objects appear only in one of a relatively small number of stable positions corresponding to the orientations that the object would take if it were thrown onto a plane. They represent one view of the object for each aspect corresponding to a stable position, and for other views in an aspect they use linear transformations of the given view.

Korn and Dyer [1985] grow regions of feature equivalence, storing one representative view of each region. Thus, the regions resemble aspects. Ikeuchi [1987] constructs and uses an "interpretation tree" in a 3-D object recognition algorithm for bin-picking. This interpretation tree is based on something similar to the aspect graph. When faces or other features are found in an image, the interpretation tree is used to select the aspect of the object with the same appearance. Castore and Crawford propose a recognition system that stores every topologically-distinct view of objects [Castore and Crawford, 1984; Crawford, 1985]. Other researchers propose that the aspect graph be used as part of a general representation for computer vision [Schneier *et al.*, 1986].

Chapter 7: Conclusion and Future Work

7-1. The Asp: A Continuous, Viewer-centered Object Representation

The asp is a new continuous, viewer-centered representation for polyhedra, the first to represent appearance as a function of viewpoint. All visual events, such as regions, edges, and T-junctions in an image, are represented as faces of the asp, and any cross-section of the asp for a fixed viewpoint is an image of the object with hidden lines removed. The asp is not a polytope since its surfaces are curved, but the general forms of the surfaces and their intersections have been presented in closed form.

Algorithms were given for the construction of the asp for convex and non-convex polyhedra, under orthographic and perspective viewing models. The asp for a convex polyhedron with **n** faces has size $\Theta(\mathbf{n})$, and it can be constructed in $\Theta(\mathbf{n})$ time. The asp for a non-convex polyhedron has size $\Omega(\mathbf{n})$ to $O(\mathbf{n}^4)$, and it can be constructed in time $O(\mathbf{mn})$, where **m** is the output size. Commonly occurring polyhedra have size closer to the convex case than the worst case. The algorithms for constructing the asp involve finding the intersection of 4-D or 5-D cells with curved surfaces.

Three applications of the asp were presented. The first is the construction of the aspect graph. After the introduction of the aspect graph by Koenderink and van Doorn [1979], its construction in the general case was an open problem until our application of the asp to the problem, resulting in the algorithm of Chapter 3. Algorithms and size bounds were given for convex and general polyhedra, under the orthographic and perspective models. Bounds on the size of the aspect graph are also bounds on the number of topologically-distinct views of polyhedra. Under the orthographic viewing model, there are $O(\mathbf{n}^2)$ aspects in the convex case and $O(\mathbf{n}^6)$ in the general case. Under the perspective model, there are $O(\mathbf{n}^3)$ aspects in the convex case and $O(\mathbf{n}^9)$ in the general case. Thus, the number of views can be very large. However, the algorithms given for constructing the aspect graph are efficient in the sense that they require time $O(\mathbf{m} \log \mathbf{m})$ for most polyhedra, where **m** is the output size.

The second application is object recognition. Chapter 4 discussed using the asp as an object representation for object recognition. Since the asp represents all visual events, it makes the problem of object recognition one of matching appearance with a representation of appearance, rather than one of matching appearance with an object-centered model. Then any feature that appears in an image can be matched with the model since every feature of appearance is represented. Since the asp explicitly represents all of the visual events associated with an object, it is particularly useful for parallel algorithms for computer vision: appearance of an object from all viewpoints can be compared with an image in parallel. The algorithm given for asp object recognition is stated in terms of parallel image feature detectors and associative memory, and with enough processors to find image features in parallel and to assign a processor to each stored object features, it performs object recognition in constant time.

The third application is animating rotation. In Chapter 5, an algorithm for animating rotation was presented that performs one to two orders of magnitude better than existing algorithms in the on-line phase, the display of line drawings of a scene with hidden lines removed as the viewpoint moves. The algorithm works by constructing the asp and the VSP as a combined data structure in a preprocessing phase. A cross-section of the asp (i.e. an image of the object) is kept and updated as the viewpoint moves. The VSP is used to make it possible to efficiently find visual events near any viewpoint. In order to reduce the space requirements, the asp and the VSP are constructed for a 1-D viewpoint space, the path of viewpoints along which views of the object are to be displayed.

7-2. When the Asp is a Useful Representation

The asp is an interesting object representation, and its use has resulted in improvements on previous solutions to some problems. It would be useful to characterize the problems for which the asp is an appropriate representation. We present here some characteristics of problems that may benefit from the using the asp.

The asp represents visual appearance, so appearance must be important to the solution of the problem. The asp represents all visual events from all viewpoints, and that can be a large amount of information. Therefore the problem must require most of the visual information or must use some of it repeatedly to justify the expense of computing the asp. Alternatively, the asp can be computed for a smaller range of viewpoints. For example, in animating rotation we computed the asp for a 1-D range of viewpoints. As a result, the amount of data computed and the computation time are reduced.

There are several ways that appearance information may be useful to a problem. A problem may require computing the appearance of a polyhedral scene from several viewpoints; in this case the asp is useful when the number of views is large enough to justify computing a data structure the size of the asp. With enough processors, the asp by itself contains all the necessary information to display views in constant time; with a single processor or a few processors, additional data (such as the VSP) is required to find the asp faces that are visible from a viewpoint quickly.

Another way that appearance information may be used is in finding regions of viewpoints of particular appearance characteristics. An example of such a problem is constructing the VSP. The VSP is a partition of viewpoint space into regions of constant aspect. Another example of such a problem is finding the viewpoints from which a particular feature or face of a polyhedral scene is visible. The asp is useful for problems involving finding regions of viewpoints of particular appearance characteristics whenever the boundaries of such a region correspond to visual events in the asp.

Some problems require determining the viewpoint at which particular visual events occur. For example, it may be desirable to find the viewpoint from which the bottom of a screw-hole is visible; from that point a screw can be inserted. In fact, object recognition can be thought of as a problem of finding the viewpoint at which an object has a particular appearance. The asp is useful for computing viewpoints at which events occur since visual events are all represented in the asp.

Finally, the asp is useful for achieving a high degree of parallelism in problems involving appearance. For example, in animating rotation, frames can be displayed in constant time with enough processors. Also, the asp can be used for object recognition to take advantage of plentiful associative memory by making viewpoint a parameter of the object representation in associative memory. Object recognition then requires fewer sequential steps.

7-3. Directions for Further Research

We believe that the asp will also prove useful for other problems. Some possibilities are computing the viewpoint or viewpoints from which other scene features are visible and tracking moving polyhedra in an image sequence. Another possible application is object recognition using moving features, that is, features with image space and viewpoint space extent as the viewer moves. The idea is that instead of matching static features such as vertices and edges with a model, it may be possible to match moving features such as vertices, edges, or regions as they change with changing viewpoint to corresponding features in the asp. This is a way of making use of the additional information available in a moving scene.

It would be interesting to construct the asp under different viewing models. In the given viewing model, all and only edges of polyhedra are visible in an image. It may be desirable to define the asp in a viewing model in which only occluding edges are visible, for example. This would be interesting for computer vision since there are fewer occluding edges, reducing the storage requirements, and occluding edges are more easily identified in images—they may be visible as light intensity edges, and they can also be computed as discontinuities in depth and in the gradient of optical flow.

The aspect representation could be extended to represent objects other than polyhedra. For example, it may be desirable to construct something like the asp for objects with curved surfaces such as generalized cylinders. To do so would require changing the viewing model so that features such as cusps and occluding contours are visible in the image. The notion of a visual event would also have to be changed to include changes in cusps and contours.

The idea of representing appearance as a function of viewpoint is new, and it appears to be a powerful tool with significant benefits for problems concerned with appearance. The asp is a representation for polyhedra of appearance as a function of viewpoint, and its use has resulted in advantageous solutions to existing problems. We expect the idea to be useful for other problems as well.

REFERENCES

Agin, G. J., "Hierarchical Representations of 3D objects," Final Report, SRI Project 1187, Stanford Research Institute, Stanford, CA, 1977.

Avis, D. and G. Toussaint, "An optimal algorithm for determining the visibility of a polygon from an edge," *IEEE Trans. Computers* **C-30**, 1981, pp. 910-914.

Badler, N. and R. Bajcsy, "Three-dimensional representations for computer graphics and computer vision," *ACM Computer Graphics* **12**(3), 1978, pp. 153-160.

Baumgart, B. G., "Winged edge polyhedron representation," report STAN-CS-320, Stanford Artificial Intelligence Lab., Stanford Univ., 1972.

Besl, P. J. and R. C. Jain, "Three-dimensional object recognition," *ACM Computing Surveys* **17**, 1985, pp. 75-145.

Binford, T. O., "Visual perception by computer," presented at *IEEE Conf. Systems and Control*, 1971.

Bolles, R. and R. Cain, "Recognizing and locating partially visible objects: the local-feature-focus method," *Int. J. Robotics Research* **1**(3), 1982, pp. 57-82.

Bolles, R. and P. Horaud, "3DPO: A three-dimensional part orientation system," *Int. J. Robotics Research* **5**(3), 1986, pp. 3-26.

Brooks, R., "Symbolic reasoning among 3D models and 2D images," *Artificial Intelligence* **17**, 1981, pp. 285-348.

Brooks, R., R. Greiner, and T. Binford, "The ACRONYM model-based vision system," *Proc. 6th Int. Conf. Artificial Intelligence*, 1979, pp. 105-113.

Canny, J., "Algorithms for model-driven mechanical parts inspection," Research Report RC 10505 (#48869), IBM Watson Research Center, 1984.

Canny, J., personal communication, 1987.

Castore, G. and C. Crawford, "From solid model to robot vision," *Proc. IEEE First Int. Conf. Robotics*, 1984, pp. 90-92.

Chakravarty, I. and H. Freeman, "Characteristic views as a basis for three-dimensional object recognition," *Proc. SPIE 336* (Robot Vision), 1982, pp. 37-45.

Chazelle, B. and L. Guibas, "Visibility and intersection problems in plane geometry," *Proc. ACM Symp. on Computational Geometry*, 1985, pp. 135-146.

Chazelle, B., L. Guibas, and D. T. Lee, "The power of geometric duality," *BIT* **25**, 1985, pp. 76-90.

Chin, R. and C. Dyer, "Model-based recognition in robot vision," *ACM Computing Surveys* **18**(1), 1986, pp. 67-108.

Clark, J., "Hierarchical geometric models for visible surface algorithms," *Comm. ACM* **19**(10), 1976, pp. 547-554.

Coons, S., "Surface patches and B-spline curves," in *Computer Aided Geometric Design*, R. E. Barnhill and R. F. Reisenfeld (Eds.), Academic Press, New York, 1974.

Crawford, C., "Aspect graphs and robot vision," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1985, pp. 382-384.

Dane, C. and R. Bajcsy, "Three-dimensional segmentation using the Gaussian image and spatial information," *IEEE Conf. Pattern Recognition and Image Processing*, Dallas, TX, 1981.

Denber, M. and P. Turner, "A differential compiler for computer animation," *ACM Computer Graphics* **20**(4), 1986, pp. 21-27.

Dévai, F., "Quadratic bounds for hidden-line elimination," *Proc. ACM Second Symp. Computational Geometry*, 1986, pp. 269-275.

Edelsbrunner, H., J. O'Rourke, and R. Seidel, "Constructing arrangements of lines and hyperplanes with applications," *SIAM J. Computing* **15**(2), 1986, pp. 341-363.

El Gindy, H. and D. Avis, "A linear algorithm for computing the visibility polygon from a point," *J. Algorithms* **2**, 1981, pp. 186-197.

El Gindy, H., "An efficient algorithm for computing the weak visibility polygon from an edge in simple polygons," Tech. Report, School of Computer Science, McGill Univ., 1984.

Ettinger, G., "Large hierarchical object recognition using libraries of parameterized model sub-parts," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1988, pp. 32-41.

Farrell, E., W. Yang, and R. Zappulla, "Animated 3D CT imaging," *IEEE Computer Graphics and Applications* **5**(12), 1985, pp. 26-32.

Fekete, G. and L. Davis, "Property spheres: a new representation for 3D object recognition," *Proc. Workshop on Computer Vision: Representation and Control*, 1984, pp. 192-201.

Fox, D. and M. Waite, *Computer Animation Primer*, McGraw-Hill, New York, 1984.

Fuchs, H., Z. M. Kedem, and B. F. Naylor, "On visible surface generation by a priori tree structures," *ACM Computer Graphics* **14**(3), 1980, pp. 124-133.

Fuchs, H., G. Abram, and E. Grant, "Near real-time shaded display of rigid objects," *ACM Computer Graphics* **17**(3), 1983, pp. 65-72.

Gigus, Z. and J. Malik, "Computing the aspect graph for line drawings of polyhedral objects," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1988, pp. 654-661.

Glassner, A., “Spacetime ray tracing for animation,” *IEEE Computer Graphics and Applications* **8**(2), 1988, pp. 60-70.

Guibas, L., J. Hershberger, D. Leven, M. Sharir, and R. Tarjan, “Linear time algorithms for visibility and shortest path problems inside simple polygons,” *Proc. Second ACM Symp. Computational Geometry*, 1986, pp. 1-13.

Hilbert, D. and S. Cohn-Vossen, *Geometry and the Imagination*, Chelsea Publishing, New York, 1952.

Holland, S., L. Rossol, and W. Ward, “CONSIGHT-1: A vision controlled robot system for transferring parts from belt conveyors,” *Computer Vision and Sensor Based Robots*, G. C. Dodd and L. Rossol (Eds), Plenum Press, New York, 1979.

Horn, B. K. P., “Extended Gaussian Images,” *Proc. IEEE* **72** (12), 1984, pp. 1656-1678.

Hubschman, H. and S. Zucker, “Frame-to-frame coherence and the hidden surface computation: constraints for a convex world,” *ACM Computer Graphics* **15**(3), 1981, pp. 45-54.

Ikeuchi, K., “Recognition of 3-D objects using the extended Gaussian image,” *Proc. 7th Int. Joint Conf. on Artifiicial Intelligence* , 1981, pp. 595-600.

Ikeuchi, K., “Precompiling a geometric model into an interpretation tree for object recognition in bin-picking tasks,” *Proc. IEEE Conf. Robotics and Automation*, 1986, pp. 321-339.

Jackins, C. L. and S. L. Tanimoto, “Oct-trees and their use in representing three-dimensional objects,” *Computer Graphics and Image Processing* **14**(3), 1980, pp. 249-270.

Kender, J. and D. Freudenstein, “What is a 'degenerate' view,” *Proc. IEEE Int. Conf. Robotics and Automation*, 1986, pp. 589-598.

Kim, H., R. C. Jain, and R. A. Volz, “Object recognition using multiple views,” *Proc. IEEE Int. Conf. Robotics and Automation*, 1985, pp. 28-33.

Koenderink, J. and A. van Doorn, “The singularities of the visual mapping,” *Biol. Cybernetics* **24**, 1976, pp. 145-176.

Koenderink, J. and A. van Doorn, “The Internal Representation of solid shape with respect to vision,” *Biol. Cybernetics* **32**, 1979, pp. 211-216.

Korn, M. and C. Dyer, “3D multiview object representations for model-based object recognition,” *Pattern Recognition* **20**, 1987, pp. 91-103.

Lavin, M. A., “An application of line labeling and other scene-analysis techniques to the problem of hidden-line removal,” Working Paper 66, Artificial Intelligence Lab., Massachusetts Institute of Technology, 1974.

Lee, D. T. and F. P. Preparata, “Location of a point in a planar subdivision and its applications,” *Proc. 8th ACM Symp. Theory of Computing*, 1976, pp. 231-235.

Levin, J. S., "Mathematical models for determining the intersections of quadric surfaces," *Computer Graphics and Image Processing* **11**, 1979, pp. 73-87.

Lieberman, L. I., "Model-driven vision for industrial automation," in *Advances in Digital Image Processing*, P. Stucki (Ed.), Plenum, New York 1979, pp. 235-246.

Lowe, D., "Three-dimensional object recognition from single two-dimensional images," Tech. Report No. 202, Courant Institute, New York Univ., 1986.

Lumia, R., M. Knapp-Cordes, and C. Witzgall, "Determining Occlusion by a polyhedral body using preprocessing," Working Report, National Bureau of Standards, Gaithersberg, MD, 1985.

March, L. and P. Steadman, *The Geometry of Environment*, MIT Press, Cambridge, MA, 1974.

Magnenat-Thalmann, N. and D. Thalmann, "An indexed bibliography of computer animation," *IEEE Computer Graphics and Applications* **5**(5), 1985, pp. 76-86.

Markowsky, G. and M. A. Wesley, "Fleshing out wire frames," *IBM J. Research and Development* **24**(1), 1980, pp. 64-74.

Marr, D., "Representing visual information," *Image Understanding Systems*, A. R. Hanson and E. M. Riseman Eds., Academic Press, New York, 1978.

Marr, D., *Vision*, W. H. Freeman and Co., San Francisco, 1982.

Marr, D. and H. K. Nishihara, "Representation and recognition of the spatial organization of three-dimensional shapes," Memo 377, Artificial Intelligence Lab., Massachusetts Institute of Technology, 1976.

McKenna, M. and R. Seidel, "Finding the optimal shadows of a convex polytope," *Proc. IEEE Symp. Computational Geometry*, 1985, pp. 90-99.

Perkins, W. A., "A model-based vision system for industrial parts," *IEEE Trans. Computers* **C-27**, 1978, pp. 126-143.

Plantinga, W. H. and C. R. Dyer, "An algorithm for constructing the aspect graph," *Proc. 27th IEEE Symp. Foundations of Computer Sci.*, 1986, pp. 123-131.

Plantinga, W. H. and C. R. Dyer, "The aspect representation," Tech. Report 683, Computer Sciences Dept., Univ. of Wisconsin–Madison, 1987.

Plantinga, W. H. and C. R. Dyer, "The asp: a continuous, viewer-centered representation for 3D object recognition," *Int. Conf. Computer Vision*, 1987, pp. 626-630.

Plantinga, W. H. and C. R. Dyer, "Construction and display algorithms for the asp," Tech. Report 735, Computer Sciences Dept., Univ. of Wisconsin–Madison, 1987.

Plantinga, W. H. and C. R. Dyer, "Visibility, occlusion, and the aspect graph," Tech. Report 736, Computer Sciences Dept., Univ. of Wisconsin–Madison, 1987.

Ponce, J., "Prism trees: an efficient representation for manipulating and displaying polyhedra with many faces," A.I. Memo 838, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1985.

Potmesil, M., "Generation of 3D surfaces from images of pattern-illuminated objects," *Proc. IEEE Conf. Pattern Recognition and Image Processing*, 1979, pp. 553-559.

Requicha, A. A. G., "Representations of rigid solids—theory, methods, and systems," *ACM Computing Surveys* **12**(4), 1980, pp. 437-464.

Requicha, A. A. G. and H. B. Voelcker, "Constructive solid geometry," Tech. Memo 25, Production Automation Project, Univ. Rochester, 1978.

Requicha, A. A. G. and H. B. Voelcker, "Solid modelling: a historical summary and contemporary assessment," *IEEE Computer Graphics and Applications* **2**(2), 1982, pp. 9-24.

Requicha, A. A. G. and H. B. Voelcker, "Solid modelling: current status and research directions," *IEEE Computer Graphics and Applications* **3**(7), 1983, pp. 25-37.

Roberts, L., "Machine perception of three-dimensional solids," *Optical and Electro-Optical Infomation Processing*, J. T. Tippett *et al.*, Eds., MIT Press, 1965.

Rogers, D. F., *Procedural Elements for Computer Graphics*, McGraw-Hill, New York, 1985.

Rosenfeld, A., "Recognizing unexpected objects: a proposed approach," *Int. J. Pattern Recognition and Artificial Intelligence* **1**(1), 1987, pp. 71-84.

Rubin, S. and T. Whitted, "A 3-dimensional representation for fast rendering of complex scenes," *ACM Computer Graphics* **14**(3), 1980, pp. 110-116.

Schneier, M., R. Lumia, and E. Kent, "Model-based strategies for high-level robot vision," *Computer Vision, Graphics, and Image Processing* **33**, 1986, pp. 293-306.

Scott, R., "Graphics and prediction from models," *Proc. Image Understanding Workshop*, 1984, pp. 98-106.

Shafer, S. A. and T. Kanade, "The theory of straight homogeneous generalized cylinders and taxonomy of generalized cylinders," Tech. Report CMU-CS-83-105, Carnegie-Mellon Univ., 1983.

Shelley, K. and D. Greenberg, "Path specification and path coherence," *ACM Computer Graphics* **16**(3), 1982, pp. 157-166.

Soroka, B. I. and R. K. Bajcsy, "A program for describing complex three-dimensional objects using generalized cylinders as primitives," *Proc. IEEE Conf. Pattern Recognition and Image Processing*, 1978, pp. 331-339.

Suri, S. and J. O'Rourke, "Worst-case optimal algorithms for constructing visibility polygons with holes," *Proc. Second ACM Symp. Computational Geometry*, 1986, pp. 14-23.

Sutherland, I., R. Sproull, and R. Schumacker, “A characterization of ten hidden-surface algorithms,” *ACM Computing Surveys* **6**, 1974, pp. 10-56.

Srihari, S. N., “Representation of three-dimensional digital images,” *ACM Computing Surveys* **13**(4), 1981.

Stewman, J. and K. Bowyer, “Aspect graphs for convex planar-faced objects,” *Proc. IEEE Int. Conf. Robotics and Automation*, 1987, pp. 123-130.

Thompson, D. and J. Mundy, “Three-dimensional model matching from an unconstrained viewpoint,” *Proc. IEEE Int. Conf. Robotics and Automation*, 1987, pp. 208-220.

Thorpe, C. and S. Shafer, “Correspondence in line drawings of multiple views of objects,” *Proc. 8th Int. Joint Conf. Artificial Intelligence*, 1983, pp. 959-965.

Tsotsos, J., “A 'complexity level' analysis of immediate vision,” *Int. J. Computer Vision* **1**(4), 1988, pp. 303-320.

Wallace, T., O. Mitchell, and K. Fukunga, “Three-dimensional shape analysis using local shape descriptors,” *IEEE Trans. Pattern Analysis and Machine Intelligence* **3**, 1981, pp. 310-323.

Wallace, T. and P. Wintz, “An efficient three-dimensional aircraft recognition algorithm using normalized Fourier descriptors,” *Computer Graphics and Image Processing* **13**, 1980, pp. 96-126.

Watts, N., “Calculating the principal views of a polyhedron,” Tech. Report 234, Dept. of Computer Science, Univ. of Rochester, 1987.

Werman, W., S. Baugher, and J. A. Gualtieri, “The visual potential: one convex polygon,” Report CAR-TR-212, Center for Automation Research, Univ. of Maryland, 1986.

Yan, J., “Advances in computer-generated imagery for flight simulation,” *IEEE Computer Graphics and Applications* **5**(8), 1985, pp. 37-51.

Zyda, J. Z., R. McGhee, R. Ross, D. Smith, and D. Streyle, “Flight simulators for under $100,000,” *IEEE Computer Graphics and Applications* **8**(1), 1988, pp. 19-27.