THE ASP: A CONTINUOUS VIEWER-CENTERED REPRESENTATION FOR 3D OBJECT RECOGNITION

by

Harry Plantinga
and
Charles Dyer

Computer Sciences Technical Report #682

January 1987

The Asp: A Continuous Viewer-Centered Representation for 3D Object Recognition

Harry Plantinga — Charles Dyer

Computer Sciences Department
1210 W. Dayton Street
University of Wisconsin - Madison
Madison, WI 53706

Abstract

In this paper we present a new representation for polyhedral objects which is a continuous generalization of multiview representations. That is, we show how to represent views or images of an object (i.e. the appearance of the object) *continuously* over *all* viewpoints. We then show how this representation overcomes some of the drawbacks of multiview representation for object recognition, namely, that the image to be recognized will usually not match one of the stored images exactly, that the representation contains no information about how to interpolate between views, and that the representation is large.

This work was supported in part by the National Science Foundation under Grant DCR-8520870.

1. Introduction

Of the many different object representations in use for 3D recognition, most are *object-centered* in the sense that they represent the volume of object space that the object occupies in some manner. Polyhedral representations such as winged-edge polyhedra [Baumgart, 1972] represent this volume by representing the boundaries of the volume. Generalized cylinders [Binford, 1971; Agin, 1977] represent the volume as the sweep of a cross-sectional area. Volumetric representations such as octrees [Jackins and Tanimoto, 1980] and constructive solid geometry [Requicha, 1980] represent the volume as a combination of simpler volumes.

However, in 3D object recognition we are concerned with the appearance of an object rather than the volume of space that it fills, and the computation of its appearance from various viewpoints is relatively time-consuming when the representation in use is object-centered. Another approach that has been explored recently addresses this problem by representing an object "in 3D" as several different 2D views of the object. This approach is often called multiview object representation. Since multiview representations represent the *appearance* of an object rather than the volume of space that it fills, they are *viewer-centered* representations.

There are some problems with this kind of approach. First, it is not very likely that any particular image of an object will exactly match one of the stored views. Another problem is the size of the representation. Researchers have proposed multiview object representations with size as large as $O(\mathbf{n}^3)$ for convex objects and $O(\mathbf{n}^6)$ for non-convex objects of **n** faces. Yet 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 nice to be able to interpolate between views, or to use the representation to suggest a new viewpoint from which to look at the object in order to get more information.

In this paper we introduce a new representation for polyhedral objects and scenes, the *aspect representation* or *asp*, which has all the advantages of multiview representations with fewer disadvantages. In this representation, we still represent the *appearance* of an object rather than the volume of space that it fills, so the representation is viewer-centered rather than object-centered. However, we represent the appearance as a continuous function of viewpoint, so that we can efficiently recover the appearance of the object from *any* viewpoint, and we can search for *viewpoints* with a particular *appearance*. And somewhat surprisingly, while the aspect representation can still be large for highly non-convex objects, it turns out to have considerably smaller size than multiview representations which attempt to represent appearance over the same range of viewpoints.

In Section 2 we discuss multiview object representation. In Section 3 we define the aspect representation and give examples of asps. We discuss various properties of asps in Section 4 and show how to represent them. In Section 5 we discuss using the aspect representation for object recognition, and in Section 6 we present algorithms for constructing asps and taking unions, intersections, and differences.

2. Multiview Object Representation

Multiview representations come in several flavors. For example, 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 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.

Koenderink and van Doorn [1979] made possible a more systematic approach to multiple view representations when they introduced the ideas of *aspect* and the *visual potential* for an object. They define *aspect* to be the topological appearance of an object from a particular viewpoint. From most viewpoints, the topological appearance of the object remains constant over small changes in viewpoint, since no faces appear or disappear. The topology remains constant even though edges may change in size and location. Thus the aspect remains constant for regions of viewpoints. Viewpoints at which the aspect changes are called *events*. The *visual potential graph* (which is also called the *aspect graph*) has vertices corresponding to these regions of viewpoints of constant aspect, and two vertices are connected if the corresponding regions of viewpoints are connected by events.

Using these ideas, researchers are able to choose viewpoints for a multiview representation in a more systematic and efficient way, since they can avoid topologically equivalent images. They can represent a single view from each of a number of aspects or one from every aspect. Chakravarty and Freeman [1982] assume that objects appear only in one of a relatively small number of stable positions and 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 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 that only by storing an image for every vertex in the aspect graph. This leads to the second problem: the size of the representation. Plantinga and Dyer [1986] have shown that the maximum number of vertices in the aspect graph is $O(\mathbf{n}^2)$ for convex objects and $O(\mathbf{n}^4)$ for general objects with $\mathbf{n}$ faces under orthographic projection—more under perspective projection. Since an image of such a scene can have size $O(\mathbf{n}^2)$ in the non-convex case, the respective worst-case sizes of a representation storing an image from every aspect are $O(\mathbf{n}^3)$ and $O(\mathbf{n}^6)$ 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 nice to be able to interpolate between views, or to use the representation to suggest a new viewpoint from which to look at the object in order to get more information.

The aspect representation is similar to multiview representations in that it represents the appearance of an object. However, it represents appearance from ranges of viewpoints continuously, so it does not have the same drawbacks as multiview representations. We can efficiently recover appearance from any viewpoint. More importantly, we can search for a *viewpoint* with a particular *appearance*. And somewhat surprisingly, while the aspect representation can still be large for highly non-convex objects, it is smaller than corresponding multiview representations. Multiview representations that store an image for every aspect have worst-case size of $O(\mathbf{n}^3)$ in the convex case and $O(\mathbf{n}^6)$ in the general case, but the aspect representation has size $O(\mathbf{n})$ in the convex case and $O(\mathbf{n}^4)$ in the general case.

Note that when we speak of *aspect* in this paper, we will use the term in a slightly different way from Koenderink and van Doorn. We define aspect as *geometric appearance*, not *topological appearance*. The difference is that by our definition, the aspect changes with every change in viewpoint: there are no regions of constant aspect. Furthermore, the term *viewer-centered* is usually used to refer to representations that represent an object with images from a discrete set of viewpoints. We extend the notion to include representing the appearance of an object from a continuous range of viewpoints. The aspect representation takes the idea of viewer-centered representation to its logical end: it represents the appearance of objects from every viewpoint as a continuous, unified whole.

3. The Aspect Representation

The aspect representation represents the appearance of an object over all viewpoints, that is, image space for all viewpoints. Thus, it represents the volume of a different space that the object fills, image space cross the space of viewpoints, which we call *aspect space*. One can picture this by picturing a video camera and a TV monitor. The coordinates of a point in image space are its coordinates on the TV monitor. Viewpoint space is the space of directions in which the camera can point. The cross product of these spaces is aspect space. The aspect representation for an object is the set of all points in aspect space that the object occupies— every point in the image of the object on the TV monitor for each camera direction.

We first define formally *viewpoints* and*viewpoint space*. In this paper we will use orthographic projection in constructing images from three-dimensional objects or scenes. Since under orthographic projection the distance from the camera to the scene is not significant, 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 will 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.

Viewpoint space is the space of points on the unit sphere. Specifically, we define the viewpoint (θ,ϕ) to be the point on the unit sphere as shown in Figure 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). For example, $(\theta=0,\phi=0)$ corresponds to looking parallel to the z-axis in the -z direction; increasing θ corresponds to points that are increasingly "to the east" on the unit sphere and increasing ϕ corresponds to points on the sphere that are further "south" when $\phi < 180°$.

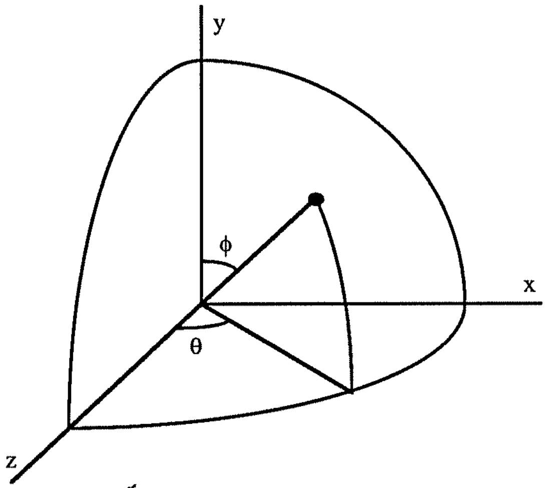

Figure 1. The Viewpoint (θ,ϕ) on the unit sphere

An image of an object is in effect 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 it by (u,v). The aspect representation for an object is a representation of the volume of *aspect space* that the object occupies, where aspect space is defined to be (θ,ϕ,u,v)— that is, the object's projection into the viewing plane (u,v) for any viewpoint (θ,ϕ). Topologically, aspect space is the cross product of a plane and a sphere, so it is a well-understood and well-behaved space. The aspect representation then represents every point $(\theta_1,\phi_1,u_1,v_1)$ of aspect space that the object occupies— that is, every point (u_1,v_1) in the image of the object from viewpoint (θ_1,ϕ_1).

In order to facilitate understanding the aspect representation, we will present some examples of asps for various objects. The first example is the asp for a point at location (x_0,y_0,z_0) in some fixed coordinate system. At $(\theta=0,\phi=0)$ the point appears at the position (x_0,y_0) in the image; as θ and ϕ vary, the point moves in the image according to the equations for rotation and projection into the image plane:

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

$$[x_0\cos\theta - z_0\sin\theta,\ x_0\sin\theta\sin\phi + y_0\cos\phi + z_0\cos\theta\sin\phi,\ 0]$$

Thus, the aspect representation of a point in 3-space is the 2-dimensional surface in aspect space given by

$$u = x_0\cos\theta - z_0\sin\theta \tag{1}$$
$$v = x_0\sin\theta\sin\phi + y_0\cos\phi + z_0\cos\theta\sin\phi \tag{2}$$

Note that in order to represent this surface, it is only necessary to store the three coordinates of the point, (x_0,y_0,z_0), since the equation of the surface is known a priori to within those three constants.

The asp for a line segment is a 3-dimensional surface in aspect space bounded by 2-surfaces. Alternatively, it can be written down directly by substituting parametric equations for a line segment into the point equations. Here 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. We let $a_1 = x_1-x_0$, $b_1 = y_1-y_0$, and $c_1 = z_1-z_0$:

$$\begin{aligned} x(s) &= x_1 + a_1 s \\ y(s) &= y_1 + b_1 s \\ z(s) &= z_1 + c_1 s \end{aligned} \tag{3}$$

The point equations then become

$$u = (x_1 + a_1 s)\cos\theta - (z_1 + c_1 s)\sin\theta \tag{4}$$
$$v = (x_1 + a_1 s)\sin\theta\sin\phi + (y_1 + b_1 s)\cos\phi + (z_1 + c_1 s)\cos\theta\sin\phi \tag{5}$$

This is a 3-dimensional surface in aspect space. The bounding 2-faces are given by Eqs. (1) and (2) above for the points (x_0,y_0,z_0) and (x_1,y_1,z_1). Note that in order to represent this surface it is only necessary to store the the coordinates of the endpoints of the line segment, (x_0,y_0,z_0) and (x_1,y_1,z_1), since the equation for the surface is known.

The asp for a triangle consists of the asps for the bounding sides and vertices as above together with connectivity information, i.e., links from the asps for bounding sides to the asps for the adjacent vertices, and so on. Figure 2 shows the asp for the triangle (1,1,0), (2,1,0), (1,2,0) in object space. Since it is difficult to represent a four-dimensional manifold in a two-dimensional figure, the figure shows several different three-dimensional cross-sections of the asp at various fixed values of ϕ. The two-dimensional cross-sections of the asp in each three-dimensional 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.

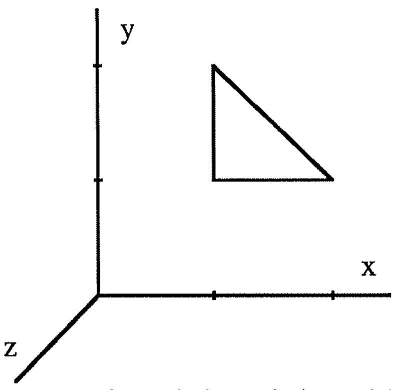

Figure 2a. The triangle (1,1,0), (2,1,0), (1,2,0) in object space.

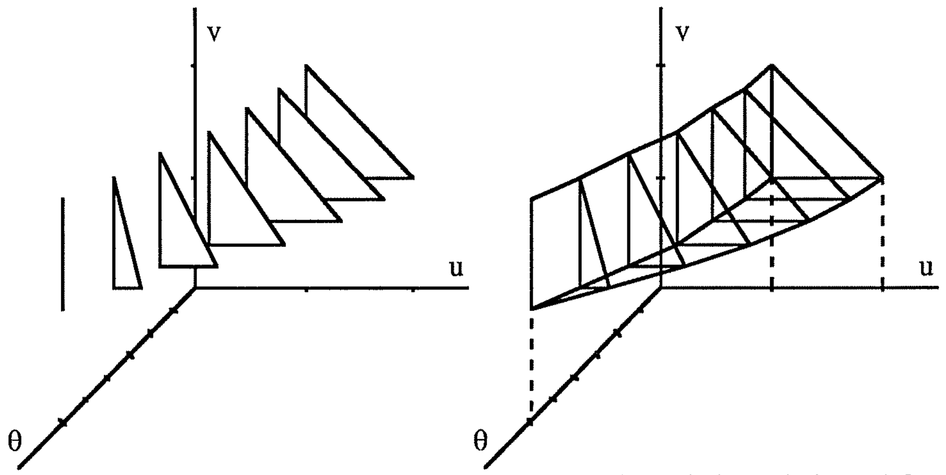

Figure 2b. A cross-section of the asp for the triangle (1,1,0), (2,1,0), (1,2,0) for $\phi = 0^\circ$.

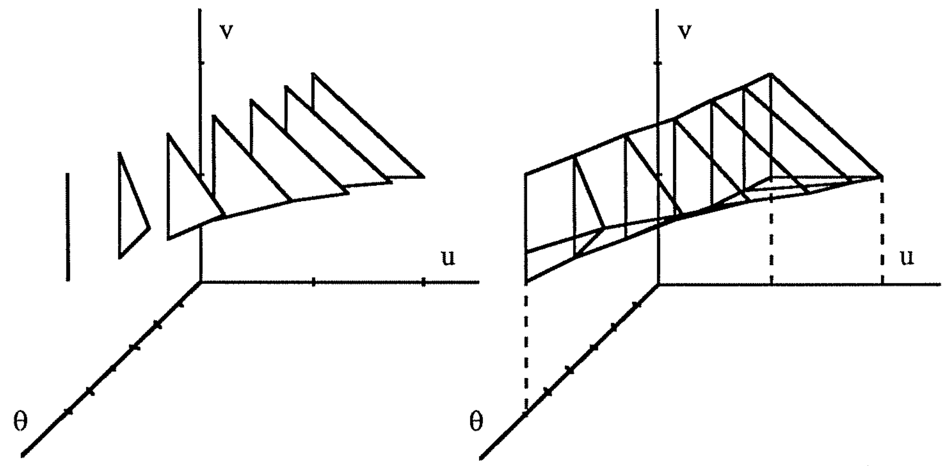

Figure 2c. A cross-section of the asp for the triangle (1,1,0), (2,1,0), (1,2,0) for $\phi = 15^\circ$.

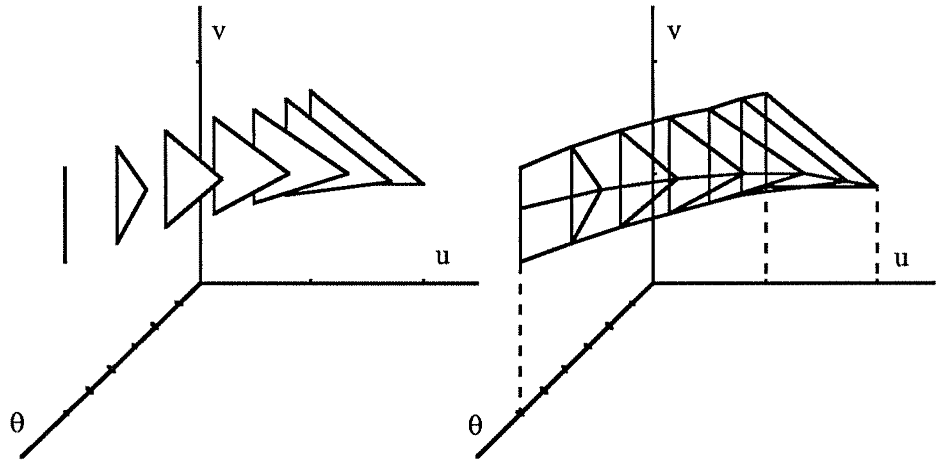

Figure 2d. A cross-section of the asp for the triangle (1,1,0), (2,1,0), (1,2,0) for $\phi = 30^\circ$.

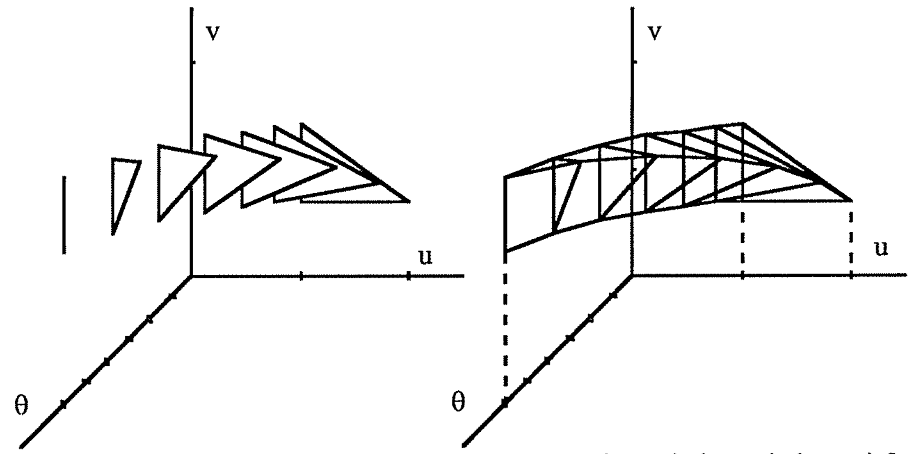

Figure 2e. A cross-section of the asp for the triangle (1,1,0), (2,1,0), (1,2,0) for $\phi = 45^\circ$.

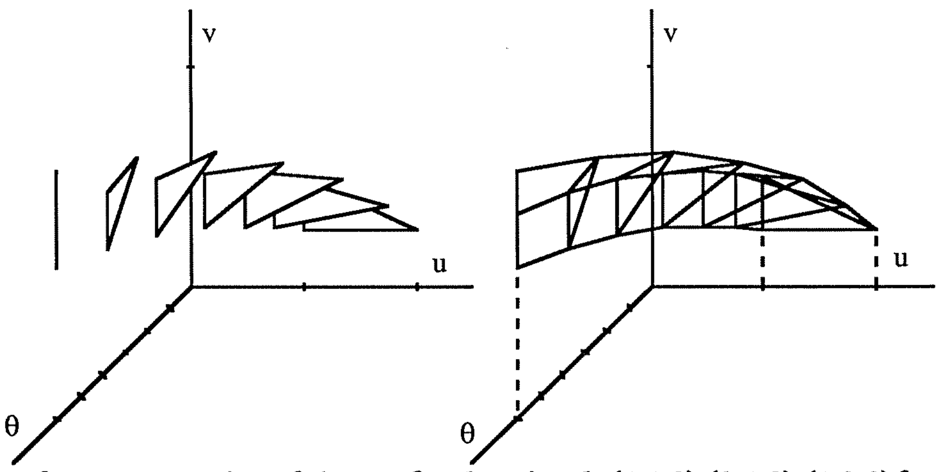

Figure 2f. A cross-section of the asp for the triangle (1,1,0), (2,1,0), (1,2,0) for $\phi = 60^\circ$.

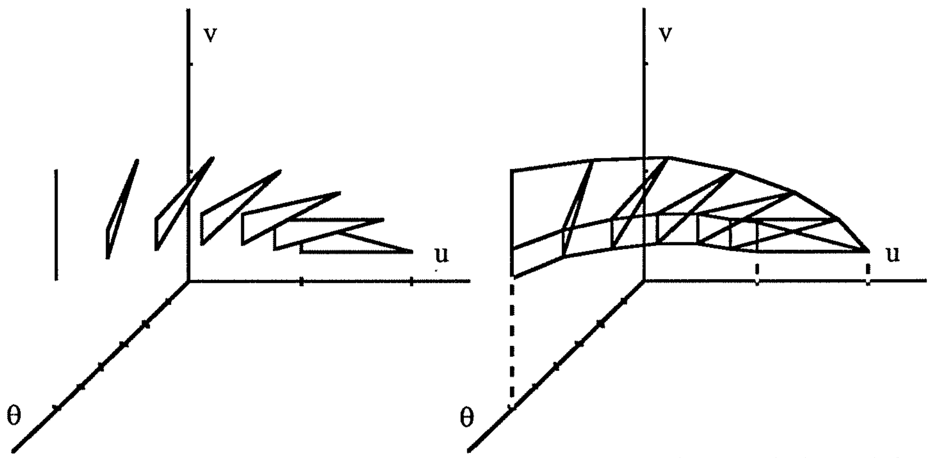

Figure 2g. A cross-section of the asp for the triangle (1,1,0), (2,1,0), (1,2,0) for $\phi = 75^\circ$.

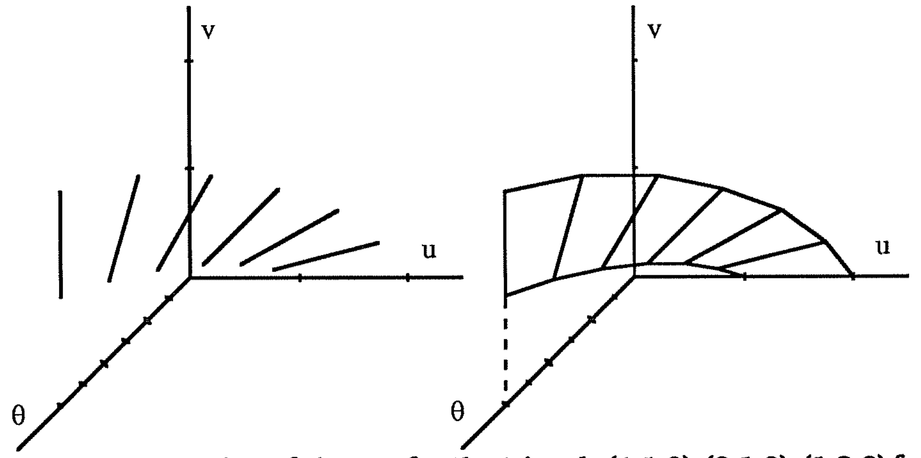

Figure 2h. A cross-section of the asp for the triangle (1,1,0), (2,1,0), (1,2,0) for $\phi = 90^\circ$.

4. Properties of Asps

In this section we discuss various properties of asps. We first discuss the structural similarity between asps and polytopes. We then show that occlusion in object space corresponds to set subtraction in aspect space. We also discuss how to represent asps and present bounds on the size of asps.

4.1. Asps as Polytopes

We represent an asp by representing the volume of aspect space that the it occupies; specifically, by representing the 3-surfaces that bound the four-dimensional cells of aspect space. Unfortunately, the surfaces are not in general planar, so the object is not a polytope. However, every (u,v)-cross section of the object is made up of polygons since the appearance of a polyhedron from any viewpoint is polygonal. Also, every arc in the projection of an asp onto viewpoint space, (θ,ϕ), is an arc of a great circle. To see this, imagine the shadow of an edge cast on a large sphere by some point light source. The shadow gets closer to the arc of a great circle as the sphere becomes larger. Thus, in the limit, the shadow is an arc of a great circle, and the projection of the asp onto (θ,ϕ) or any (θ,ϕ)-cross section of the asp is spherical polygonal.

For these and other reasons, the surfaces that arise in the aspect representation for a polyhedon are well-behaved, and we can work with them in much the same manner as we would work with lines and planes in a polyhedron. For example, there is only one surface of intersection of two aspect surfaces in which we are interested (and its polar opposite), and we can easily calculate what it is. We show how to work with these surfaces below. Thus, while the asp is not a polytope, we will speak of it as if it were. Specifically, we will refer to the 3-surfaces that bound the asp as "facets," a term one would normally use for the 3-dimensional faces bounding a 4-polytope. We will refer to the 2-surfaces that bound the facets as "ridges" and the 1-surfaces or curves as "edges." A 4-dimensional volume of aspect space bounded by facets will be referred to as a "cell." In general, the asp will consist of a region or regions of aspect space partitioned into cells; these cells correspond to polygons in the image of the object and the facets bounding the cells correspond to edges.

Thus, the aspect representation is very much like a polyhedral representation except that we must store the equation for each "face," since it is a curved surface. We will show below how to represent the asp with a data structure similar to that of a polyhedron, except of course that the asp is four-dimensional rather than three and the four-dimensional volume of the asp is partitioned into four-dimensional cells. In this manner, we can work with asps much as we would with polyhedra. For example, we can determine whether two asps intersect by determining whether some face of one intersects some face of the other, and if not, whether one is completely inside the other.

4.2. Asps and Visibility

Consider again the asp for a point. Eqs. (1) and (2) for the asp for that point are defined for all values of θ and ϕ, as one would expect, since a point is normally visible from any viewing direction. But we can also represent a point which is not visible from every viewing direction, say a point which is occluded from some directions by a polygon, by putting bounds on the values of θ and ϕ for Eqs. (1) and (2) 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 aspect representation for a point which is visible from only some viewpoints.

In fact, since the aspect representation for an object represents the volume of aspect space that the object occupies, if a polygon $\mathbf{p}_2$ is in front of the plane containing another polygon $\mathbf{p}_1$, the set subtraction of the asp for $\mathbf{p}_2$ from the asp for $\mathbf{p}_1$ represents the volume of aspect space where $\mathbf{p}_1$ is visible, i.e. not occluded by $\mathbf{p}_2$. That is, the point $(\theta_1,\phi_1,u_1,v_1)$ is in the asp whenever (u_1,v_1) is a point in the image of $\mathbf{p}_1$ from viewpoint (θ_1,ϕ_1) but not in the image of $\mathbf{p}_2$. We will call this the asp for $\mathbf{p}_1$ *as obstructed by* $\mathbf{p}_2$. In Figure 3 we show the asps for two triangles in object space, $\mathbf{p}_1$ and $\mathbf{p}_2$. The difference between $\mathbf{p}_1$ and $\mathbf{p}_2$ is that $\mathbf{p}_1$ (on the left) is in the z=0-plane, while $\mathbf{p}_2$ is in the z=1-plane. Figure 4 shows the subtraction of the asp for $\mathbf{p}_1$ from the asp for $\mathbf{p}_2$, which is the asp for $\mathbf{p}_1$ as obstructed by $\mathbf{p}_2$.

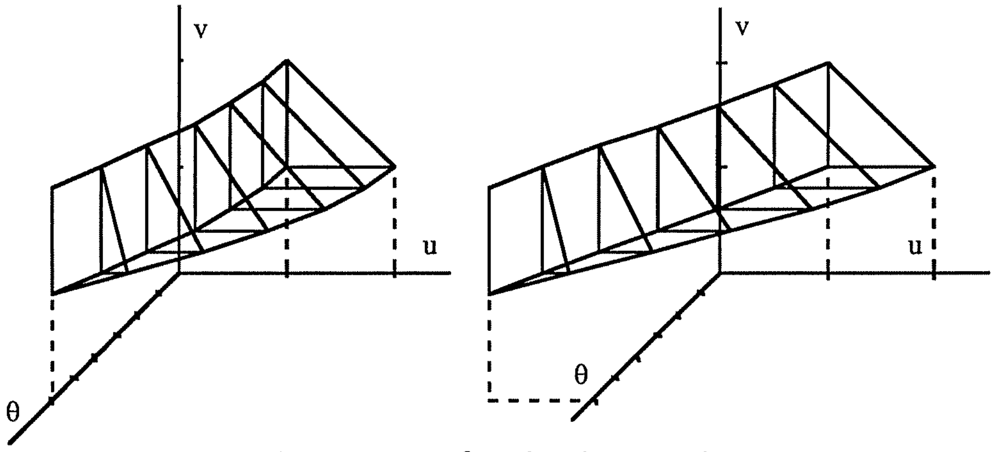

Figure 3. Asps for triangles p_1 and p_2.

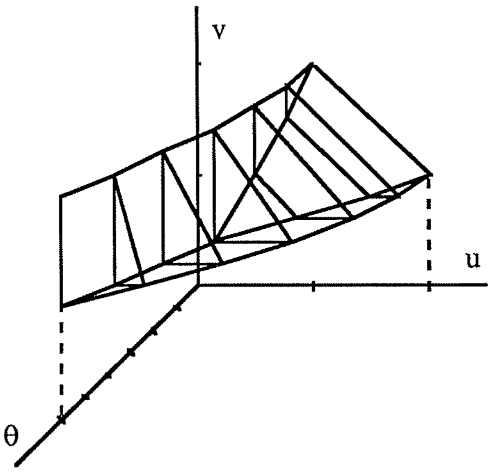

Figure 4. The subtraction of the asp for p_1 from the asp for p_2, which is the asp for p_1 as obstructed by p_2.

Note that if we take the cross-section of the asp for $\mathbf{p}_1$ as obstructed by $\mathbf{p}_2$ for a fixed value of (θ,ϕ), we get the image of $\mathbf{p}_1$ as obstructed by $\mathbf{p}_2$ at that viewpoint—namely, that part of $\mathbf{p}_1$ which is not behind $\mathbf{p}_2$. Therefore the cross sections of the asp in Figure 4 are the visible parts of $\mathbf{p}_1$. Also, if we take the projection of the asp for $\mathbf{p}_1$ as obstructed 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, we can find the region of viewpoint space in which some face $\mathbf{f}$ of the polyhedron is visible by subtracting the asp for every other face from the asp for $\mathbf{f}$ 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 in the proper manner, 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 can pre-compute hidden-line removal information for a scene from all viewpoints.

In order to better understand the aspect representation, we will point out some relationships between features of the appearance of the object and features of the asp. Imagine looking at a feature of the object and varying the viewpoint slightly, noting what happens to the feature in the image of the object. By noting what happens to the feature we can determine the dimension of the corresponding asp feature. For example, suppose two vertices of a polyhedron appear at the same point in the image from some particular viewpoint. If the viewpoint is changed slightly in any direction, the "feature" will disappear, that is, the two vertices will no longer appear at the same image point. Thus the asp feature that corresponds to a pair of coincident points in an image is zero-dimensional— it has no extent in viewpoint space. Since the feature is also zero-dimensional in the image, the resulting feature of the asp is zero-dimensional, i.e. a vertex.

Suppose that from some viewpoint two edges of the polyhedron appear to intersect. If the viewpoint changes in any direction by a small amount, the edges will still appear to intersect in general, although perhaps at a different location in the image. Then the feature (the point of intersection in the image of the two lines) has extent in both θ and ϕ. The feature is zero-dimensional in the image, so the corresponding feature of the asp is a 2-dimensional surface or a ridge.

With considerations such as this in mind, we present a table of features in the image of the object and corresponding features in the asp (see Table 1). For example, if four edges of the object appear to intersect in some image, then there is a corresponding vertex in the asp.

Image feature	Asp feature
point	ridge (2D)
edge	facet (3D)
polygon	cell (4D)
apparent intersection point of two edges	ridge
apparent intersection of edge and vertex	edge (1D)
apparent intersection of three edges	edge
apparent intersection of two vertices	vertex
apparent intersection of two edges and a vertex	vertex
apparent intersection of four edges	vertex

Table 1. Image features and corresponding asp features

4.3 Aspect Surfaces

In this section we characterize the surfaces that arise in aspect representations and show how to represent them efficiently. We also show how to find intersections of pairs of such surfaces and describe a data structure that can be used to represent asps.

We derived in Section 3.1 the equations for a 3-dimensional aspect surface corresponding to an edge in the polyhedron. With t_1 substituted for s, they are:

$$u = (x_1 + a_1 t_1) \cos\theta - (z_1 + c_1 t_1) \sin\theta \qquad (6)$$
$$v = (x_1 + a_1 t_1) \sin\theta \sin\phi + (y_1 + b_1 t_1) \cos\phi + (z_1 + c_1 t_1) \cos\theta \sin\phi \qquad (7)$$

This is the general form of the 3-surfaces that arise in aspect representations for polyhedral objects. Note that since the form of the surface is known *a priori*, in order to represent the surface it is only necessary to store the six constants in the equations—x_1, y_1, z_1, a_1, b_1, and c_1. Also, since $a_1 = x_2 - x_1$, $b_1 = y_2 - y_1$, and $c_1 = z_2 - z_1$, it is in fact only necessary to store the endpoints of a line segment in order to characterize the corresponding aspect 3-surface.

Thus, we can represent a line in object space and a corresponding 3-surface in aspect space with the same six constants; however, a facet of the asp is only a subset of that 3-surface bounded by 2-surfaces just as a polygon is a portion of a plane bounded by edges. In order to represent a facet, we must represent the 3-surface on which it lies and the 2-surfaces that bound it. The boundaries of a facet occur at the intersection of two 3-surfaces, so in order to store a facet we must store the ridges that bound it, and in order to do that, we must characterize the types of 2-surfaces that arise.

Since 2-surfaces arise as the intersection of two 3-surfaces, we must determine the intersection of two general 3-dimensional aspect surfaces. Let the first be given by Eqs. (6) and (7) above and the second by

$$u = (x_2 + a_2 t_2) \cos\theta - (z_2 + c_2 t_2) \sin\theta \qquad (8)$$
$$v = (x_2 + a_2 t_2) \sin\theta \sin\phi + (y_2 + b_2 t_2) \cos\phi + (z_2 + c_2 t_2) \cos\theta \sin\phi \qquad (9)$$

Solving these four equations for t_1 and eliminating u, v, and t_2, we get:

$$t_1 = \frac{d_1 \sin\theta + d_2 \cos\theta + d_3 \tan\phi}{d_4 \sin\theta + d_5 \cos\theta + d_6 \tan\phi} \qquad (10)$$

where

$$\begin{aligned} d1 &= c2\ (y2 - y1) - b2\ (z2 - z1) \\ d2 &= b2\ (x2 - x1) - a2\ (y2 - y1) \\ d3 &= c2\ (x2 - x1) - a2\ (z2 - z1) \\ d4 &= b1\ c2 - b2\ c1 \\ d5 &= a1\ b2 - a2\ b1 \\ d6 &= a1\ c2 - a2\ c1 \end{aligned}$$

Now we can get the equations for a general 2-surface in the aspect representation of a polyhedron by substituting Eq. (10) into Eqs. (6) and (7). Note that twelve constants characterize a general 2-surface of the aspect representation for a polyhedron. A sufficient set of twelve constants to characterize an asp 2-surface is x_1, y_1, z_1, a_1, b_1, c_1, $d_1, \ldots, d_6$. However, we can get all of these constants from the endpoints of the two line segments that gave rise to the asp, so in order to represent an asp 2-surface we need only store pointers to the two object edges that give rise to the 2-surface.

One-dimensional surfaces or curves in the aspect representation arise as the intersection of three 3-surfaces or a 3-surface and a 2-surface. Since a 2-surface arises as the intersection of two 3-surfaces, these cases are equivalent. In order to characterize the types of 1-surfaces that arise, then, it is sufficient to find the intersection of a general 3-surface and 2-surface. Again, the three object edges that give rise to the 1-surface characterize it; it is sufficient to store pointers to the three edges. However, in order to find unions and intersections of asps, we will need to determine things like whether a point is in an edge on a curve; to solve problems like this we need an explicit parametric representation of the curve. We can calculate this representation for a general aspect curve in closed form, but the result is somewhat long and complicated. Instead, we present a procedure for obtaining this representation in specific cases.

In order to derive a procedure for finding a parametric representation of an aspect 1-surface or curve, recall that they arise as the intersection of three 3-surfaces in aspect space. The intersection of three 3-surfaces in aspect space is equivalent to the apparent intersection of three line segments in image space, which in turn is equivalent to being able to draw a line through three line segments in object space. Thus we can characterize a 1-surface in aspect space by characterizing all of the lines through three edges in the object, $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$. We calculate this by postulating a viewing line $\mathbf{l}$ through the three object edges. We will assume that $\mathbf{l}$ is represented parametrically in a parameter $\mathbf{s}$ and that $\mathbf{l}$ intersects $\mathbf{e}_1$ at $\mathbf{s}=0$ and $\mathbf{e}_2$ at $\mathbf{s}=1$. The result is a set of 9 equations in 10 unknowns. From there it is easy to solve for $\mathbf{l}$ in terms of the parameter of $\mathbf{e}_3$ using the quadratic equation. From $\mathbf{l}$ it is easy to calculate (θ,ϕ,u,v) in terms of that parameter.

Finally, a vertex in the asp results from the intersection of four 3-surfaces, two 3-surfaces and a 2-surface, two 2-surfaces, or a 3-surface and a 1-surface. Again, it is sufficient to represent the vertex with pointers to the generating edges of the object. We can calculate the vertex explicitly using a procedure similar to that for the curve case above; the only difference is that we must find a line through four object edges instead of three. The result is a point of aspect space, and the solution requires the use of the cubic equation.

In summary, we can represent the surfaces in the aspect representation for an object according to how they arise. For example, asp 3-surfaces arise from edges of the object; thus, we can represent them with pointers to the appropriate edge. Similarly, a 2-surface arises as the intersection of two 3-surfaces; thus we can represent a 2-surface by pointers to the two object edges. In working with faces of asps, we will need explicit parametric representations of the surfaces. We have presented these for 3- and 2-surfaces, and we have presented procedures for calculating explicit representations of curves and surfaces.

Note that the surfaces characterized here can be calculated by solving polynomials of at most third order, without any transcendental functions. The transcendental functions arise in converting from a parametric set of lines which represent viewing directions to the (θ,ϕ,u,v) representation. This is desirable because polynomials are easier to work with than complicated transcendental functions.

4.4. Representing Asps

We can represent an asp in a hierarchical structure similar to a polyhedron or polytope. We can represent a polyhedral subdivision of E^3 as a list of cells or polyhedra, each polyhedron as pointers to bounding faces, each face as pointers to bounding edges, and each edge as a pair of bounding points. In a similar way, we can represent an asp as a list of cells of aspect space, each cell by pointers to the bounding facets, each facet by pointers to the bounding ridges, each ridge by pointers to the bounding edges, and each edge by the two bounding vertices.

In the case of the polyhedron, the line on which a line segment lies is stored implicitly. It is the line which intersects both endpoints. Similarly, it is not necessary to represent the plane on which a face lies since it is the plane defined by any three endpoints of edges bounding the face. This is also true of the aspect representation when aspect surfaces are represented by pointers to the edges of the object which give rise to the aspect surfaces. For example, suppose two endpoints of an edge of an asp are each represented by pointers to the four object edges which give rise to that point. The two points must have three of their four generating object edges in common if they lie on the same 1-surface, and these three edges define the 1-surface on which the curve lies. Similarly, two edges bounding a ridge have two of their three generating object-edges in common; these two edges define the surface that the ridge lies on.

In order to construct efficient algorithms for dealing with asps, however, we want quicker access to adjacency information in the asp than that possible in the data structure defined so far. For example, we want to be able to find the two cells separated by a facet in constant time. In order to be able to retrieve adjacency information efficiently, we make the pointers in the data structure bidirectional. Cells have pointers to the bounding facets and facets have pointers to the cells they separate. Facets have pointers to the bounding ridges and ridges have pointers to the incident facets. Ridges have pointers to bounding edges, edges to bounding vertices, and vice versa.

In a four dimensional space, a data structure like this is not necessarily of linear size in the number of faces of a polytope as it would be in three dimensions. However, in the case of the aspect representation, it turns out that it does have linear size because of the close connection with a polyhedron [Plantinga & Dyer, 1987].

4.5. Size of Asps

In the case of a convex polyhedron, the asp has size O(**n**) for an object with **n** faces, since there is a face in the polyhedron for every facet in the asp, and the size of the facets in the asp correspond directly to the size of the faces in the polyhedron. However, 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. Thus we have an immediate lower bound of $\Omega(\mathbf{n}^2)$ size, since the image of a scene (or polyhedron) with hidden lines removed can have $\Omega(\mathbf{n}^2)$ line segments. Just how much bigger than $O(\mathbf{n}^2)$ can the asp be?

In order to bound the size of a four-dimensional polytope, it is not sufficient to bound the number of vertices or facets. 4-polytopes that have **n** vertices can have $O(\mathbf{n}^2)$ 2-faces. So let us determine a bound on the number of 2-faces that the asp can have for a polyhedral object with **n** vertices. 2-faces of the asp result from two-dimensional features of the appearance—two-dimensional image features visible at a single viewpoint, one-dimensional appearance features visible over a one-dimensional range of viewpoints, or zero-dimensional image features visible over a range of viewpoints. Since there are more zero-dimensional appearance features than one- or two-dimensional ones, we will restrict our attention to them. The different possible zero-dimensional features are vertices, the apparent intersection of a vertex and an edge, and the apparent intersection of two edges. There are thus $O(\mathbf{n}^2)$ different features from which asp 2-faces can arise.

However, each image feature may give rise to a number of different asp 2-faces if it is visible in a number of different regions of viewpoint space. In how many different regions of viewpoint space can a feature be visible? Consider what bounds these regions. In the case of two edges which appear to intersect, the boundary is the line of viewpoints such that the endpoint of one of the edges appears to lie on the other edge. In the case of the apparent intersection of a vertex and an edge, the boundaries occur at the viewpoint where the vertex "lines up with" the endpoints of the edge. In all, there are $O(\mathbf{n}^2)$ different possible boundaries. Therefore, there are at most $O(\mathbf{n}^4)$

2-faces in the asp and its total size is O($\mathbf{n}^4$).

It is easier to find bounds on the number of vertices in the asp, however. A vertex is formed by the apparent intersection of four edges in an image, so there are O($\mathbf{n}^4$) vertices in the asp. It is possible to determine various other bounds on the number of vertices by examing Table 1 above. In Table 1 we see that there are actually three different ways in which a vertex in the asp can occur, i.e. when four different edges appear to intersect in the same point: when two vertices in the polyhedron appear at the same point in the image, when a vertex and two unconnected edges appear at the same point in an image, and when four unconnected edges appear to intersect in the same point. 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 in the same point is O($\mathbf{n}^3$), and the number of ways four edges can appear to intersect in the same point is O($\mathbf{n}^4$). Thus, if the object is simple enough that the number of ways four unconnected edges can appear to intersect in the same point is bounded by O($\mathbf{n}^3$), then we can immediately say that the size of the asp for that object is bounded by O($\mathbf{n}^3$). Also, if the number of ways two edges and a point can appear to intersect in a point is bounded by O($\mathbf{n}^2$), then the size of the asp for that object is bounded by O($\mathbf{n}^2$). These bounds are summarized in Table 2.

type of polyhedron	bound on size of asp
general	O($\mathbf{n}^4$)
number of ways four edges can appear to intersect in a point is bounded by O($\mathbf{n}^3$) (e.g. no grid behind a grid, but a picket fence behind a grid is acceptable)	O($\mathbf{n}^3$)
number of ways two edges and a vertex can appear to intersect in a point is bounded by O($\mathbf{n}^2$) (e.g. no picket fence behind a grid, but a grid is acceptable)	O($\mathbf{n}^2$)
convex	O($\mathbf{n}$)

Table 2. Bounds on the size of the asp

Some of these restrictions are not at all unrealistic. For example, the only way to get an asp of size Ω($\mathbf{n}^4$) 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 5). This sort of situation doesn't occur often in common images. Note also that in images with many faces, the reason for the large number of faces is often that they are simulating 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.

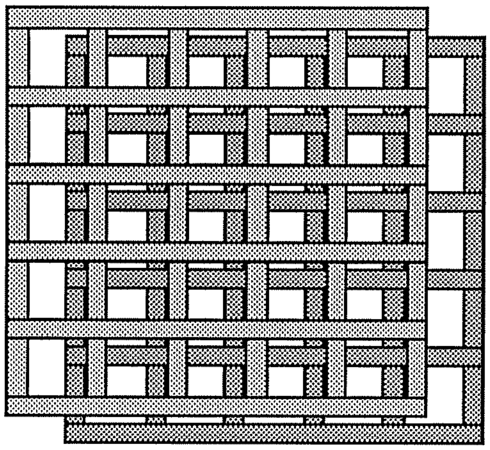

Figure 5. An example of an object for which the asp has size O(n^4).

5. Using the Aspect Representation for Object Recognition

At first it may seem as though the major benefit of the aspect representation over multiview representations for object recognition is that it represents the appearance of the object from all viewpoints instead of a discrete few. That is indeed a benefit; matching an image to one of the views in a multiview representation will have to tolerate more error than matching to the asp, since the test image will generally not match any of the stored images exactly.

However, there are other important benefits. The standard way of using a multiview representation for object recognition is to compare each stored image with the image to be recognized. In effect, this amounts to hypothesizing a viewpoint and testing whether the given image is an image of the object from the hypothesized viewpoint. This is very time-consuming, and worse still, chances are we will never guess exactly the right viewpoint.

The aspect representation suggests a different approach. Rather than hypothesizing a viewpoint and checking whether the images match, we hypothesize a match of features of the image and the asp and calculate the viewpoints from which that match would occur, if any. For example, suppose that the object we are trying to recognize will not be occluded by anything else, although there may be self-occlusion. Consider a pair of intersecting lines in a given image of the object, i.e. a corner together with edge lengths. This image feature corresponds to two facets of the asp intersecting at a ridge.

Hypothesize a match of the corner in the object to all similar features of the asp gives us enough information to calculate the viewpoints where the image would have the given feature, if any. If there are any such viewpoints, there will only be one or a small constant number of them, not a continuous range of viewpoints. If there is a possible viewpoint, we can test it by finding other nearby features in the asp that should appear in the image from that viewpoint, and checking whether they do in fact appear.

This object recognition algorithm is more precise than algorithms for multiview representations, since it looks for exact matches rather than close matches or topologically equivalent images. It also requires less runtime than algorithms for multiview representations, since it only compares a corner of the object with all corners of the asp, rather than comparing an image of an object with all of the views of a multiview representation. There are fewer corners of an asp than topologically distinct images of an object and comparing two corners takes less time than comparing two images.

The benefit of using the asp for an algorithm like this over using a standard polyhedral representation is that self-occlusion is automatically taken care of. That is, if we were to redesign this algorithm to use a standard polyhedral representation, edge length information would be of less value since the image edge may be only part of an object edge due to self occlusion. However, the asp has occlusion edges built in. For objects with no self occlusion, the algorithms would be the same.

Another benefit of the aspect representation over multiview representations is size. The asp for an object over a range of viewpoints is smaller that multiview representations which store all topologically distinct images of the object over the same range of viewpoints. It may seem counterintuitive that the asp should be smaller when it stores the appearance of the object from *all* viewpoints, but the difference is one of efficiency. In the multiview representation, a whole new image is necessary to represent one topological change in the appearance of the object. In the aspect representation, just the change is represented.

In the convex case, the size of the asp for an object is linear in the size of the object. However, since there are $O(\mathbf{n}^2)$ topologically distinct views of a convex object [Plantinga and Dyer, 1986b], the multiview representation may require as much as $O(\mathbf{n}^3)$ space. In the non-convex case, there are $O(\mathbf{n}^4)$ topologically distinct views of an object in the worst case. Furthermore, each view can have size $O(\mathbf{n}^2)$. Thus the size of a multiview representation for a non-convex object can be as much as $O(\mathbf{n}^6)$. The size of the asp for an object is $O(\mathbf{n}^4)$ in the worst case. While this is better than the multiview representation, it is still a bad worst-case. However, the worst case is only realized for highly non-convex objects; most natural objects are visually much less complex.

Another problem with multiview representations is that there is generally no information stored about how the different views of the object relate to one another. They might as well be views of different objects. This fact makes it impossible to use multiview representations to predict what will happen to the image if we vary the viewpoint slightly or suggest a new, nearby viewpoint with a certain property, say, that a certain feature is visible. This kind of computation is easily done with the aspect representation.

6. Algorithms for Asps

In this section we present algorithms for working with asps. The first algorithm constructs the aspect representation for polyhedral objects. Then we present algorithms for the intersection, union, and difference of asps, which are useful for constructing the volume of aspect space in which two features of an object are both visible at the same point, one or the other is visible at some point, or one in particular is visible.

6.1. Constructing the Asp

We have already shown the aspect representation for a line segment in Eqs. (4) and (5) above. In order to construct the asp for a polygonal face, we need only construct the asps for the edges of the face and the vertices, and add connectivity information. That is, we use a data structure of asps for line segments which specifies adjacency relationships.

We construct the aspect representation for more complex objects, such as polyhedra or polyhedral scenes, out of the asps for the faces. However, we must be careful about faces which overlap in the image plane from some viewpoint. One of these faces will be in front of the other one and thus occlude it; therefore the asp for the other one should be reduced in aspect volume appropriately to reflect the fact that parts of the face are not visible from some viewpoints.

We do this for each face individually. For some face **f** of the polyhedron we first find all of the faces that are in front of it. That is, we find the faces that are on the outside of the plane containing **f**. Some faces may cut the plane of **f**, and these we cut into two pieces (or as many pieces as necessary) at the intersection line with the plane of **f**. Each of these faces and face pieces obstructs the view of **f** from some viewpoints; the faces behind the plane containing **f** do not obstruct the view of **f** from any viewpoint.

We find the asp for the face **f** as obstructed by the other faces by taking the volume of aspect space occupied by the asp for **f** and subtracting the volume of aspect space occupied by the asps for the faces in front of **f**. Subtraction is the appropriate operation here since for the viewpoints in which the images of the object overlap, the other faces will be in front of **f** and will thus be obstructing the view of **f**. We present the algorithm for subtracting one asp from another below.

The asp for a polyhedron or a set of polyhedra is the union of the asps for the faces. In other words, we construct the asp for a face by subtracting the asps for all of the faces in front of it, and we construct the asp for a polyhedron by taking the union of the asps for the faces.

6.2. Finding the Difference of Asps

Given the asp for two objects, the difference of these asps is the region of aspect space where the second asp does not intersect the first. For any cross section for a given value of (θ,ϕ), then, the difference is the region of the image plane where the image of the second object does not overlap the image of the first. If we knew that the second object were in front of the first, so that it would occlude the first at any point where they overlapped, then the difference would be the region of aspect space where the first object is visible (i.e. not occluded by the second). This operation is important in the construction of the asp for polyhedral objects since constructing the asp involves subtracting the asps for all of the faces in front of a given face **f** from the asp for **f**.

We find the difference of two asps by taking the intersection of the first asp with the complement of the second. The complement of an asp is the asp with inside/outside information reversed. We give an algorithm for finding the intersection of asps below.

6.3. Finding the Intersection of Asps

The intersection of two asps corresponds to the region of aspect space where both objects occur at the same point in the image. For example, if we construct the aspect representations for two features of an object and take their intersection, we are left with the region of aspect space where they appear to overlap. All of the algorithms we describe for working with asps are based on the intersection algorithm described here. Note that this algorithm is defined for "polytopes" in aspect space, not subdivisions, i.e. a single cell (possibly with holes).

The algorithm for constructing the intersection of two asps is essentially the standard naive algorithm for constructing intersections. It involves testing every facet from one of the asps for intersection with the other asp. Whenever we find an intersection, we cut the facet into pieces along the surface of intersection so that we are left with two faces connected by ridges. We repeat this procedure for every facet of one of the asps and then for every facet of the other asp. When we have finished, we join equal faces in the two asps. That is, we modify our data structures so that the faces are shared by both asps. See Figures 6 and 7 for the two-dimensional analog.

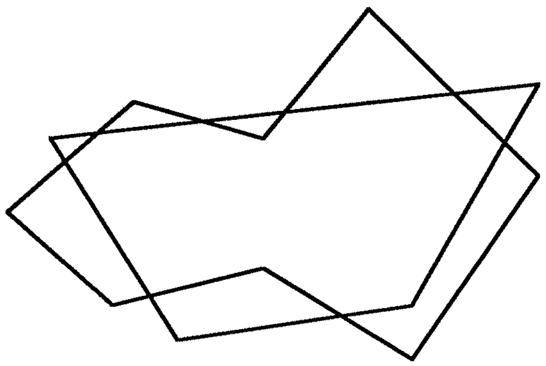

Figure 6. Overlapping polygons

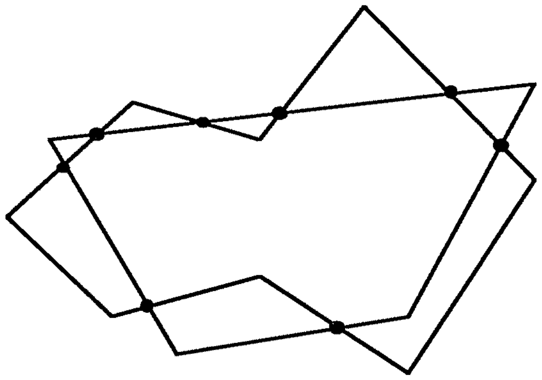

Figure 7. Finding the intersection points

When we are done with this step, we are left with an asp-like data structure which contains both asps and the common intersection points, edges, and ridges. We then "cut away the outer faces" from this data structure. That is, whenever facets of both asps meet at a ridge, we cut away the outer ones and leave the inner ones. We can determine which are the outer ones and which are the inner ones since we know which side of each facet was outside the asp and which was inside. What remains after cutting away the outer faces is the intersection of the two asps. See Figure 8 for the two-dimensional analog.

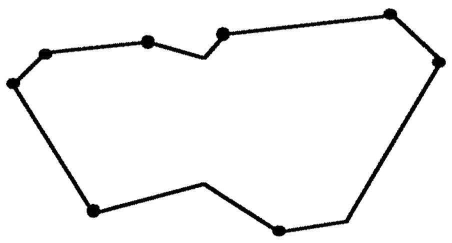

Figure 8. Cutting away the outer structure leaves the intersection

This discussion assumes that we know how to find the intersection of pairs of faces of the asps in order to find the intersection of the whole asps. It is not immediately obvious how to find these intersections, however. In fact, the problem is very similar to the original problem, except that the dimension of the problem is less. Therefore, we will consider how to solve lower-dimensional intersection problems first.

In any intersection problem, we first find the intersection of the spaces in which the faces lie. For example, if we want to find the intersection of a ridge and a facet in aspect space, we first find the intersection of the 2-surface and 3-surface containing the respective faces, since the intersection must lie in that subspace. Once we have found the intersection, we must find the part of each face that lies in it. Finding the part of a face in a subspace is a lower-dimensional intersection problem and can be handled recursively. In the example above, the intersection will be a 1-surface in general.

Next, we determine the intersection of the faces according to the dimension of the problem. If the resulting space is 1-dimensional, we have one of three cases: two points on a curve, a point and a segment on a curve, and two segments on a curve. In order to determine the intersection, we need to be able to determine ordering on the curve. We do this by calculating the parametric representation for the curve as discussed in Section 4.3 and finding the value of the parameter for the points in question. The solutions are then obvious.

If intersection of the spaces containing the faces is 2-dimensional, we then have an intersection of two 0-, 1-, or 2-dimensional faces. If the faces are not both 2-dimensional, we again find the intersections of the surfaces containing the faces and solve the lower-dimensional intersection problem. If they are both 2-dimenisonal, we find the intersection of every pair of edges of the faces, and where faces intersect we split the edges and add vertices as in Figures 5-7 above. We find the resulting intersection by "cutting away the outer structure," that is, looking for vertices with four adjacent edges and removing the outer ones.

If the intersection of the spaces containing the faces is 3-dimensional, we again check to make sure that both faces are 3-dimensional; if not, we find the intersection of the new containing spaces and solve the lower-dimensional problem. If both faces are 3-dimensional, we find the intersection of each 2-face bounding one of the 3-faces with the other 3-face and split the 2-faces accordingly. We repeat the procedure for all of the 2-faces of both 3-faces, and then join equal faces of the two 3-faces. We then cut away the outer faces; what results is the intersection.

6.4. Finding the Union of Asps

The union of two asps corresponds to the region of aspect space where either one or both of the objects are visible at some point in the image. For example, if we construct the aspect representations for two features of an object and take their union, we are left with the region of aspect space where at least one of the features appears.

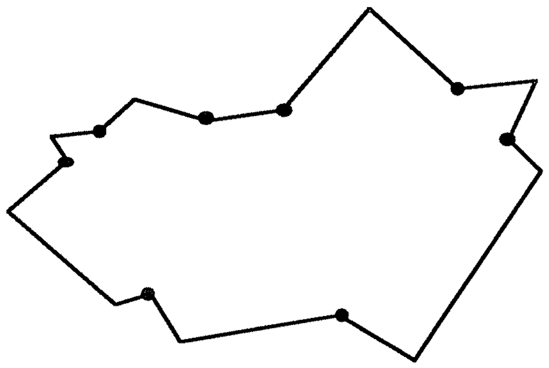

Figure 9. Cutting away the inner structure leaves the union

The union of two asps is constructed exactly the same way that the intersection is constructed, except that during the "cutting away" step we discard the inner faces rather than the outer ones. See Figure 9 for the two-dimensional analog.

6.5. Algorithm Runtimes

All of these algorithms are based on the intersection algorithm and require the same runtime. In the case of the intersection algorithm, the first step involves finding the intersection of each face of one asp with all of the faces of the other asp. This requires a total of O(**nm**) time for two asps with **n** and **m** faces respectively. The second step is the "cutting away process," which involves counting the number of facets at each ridge and deleting the external ones. Since this involves checking each ridge-facet incidence at most once, and since the number of facets at a ridge is constant, this step takes time linear in the size of the intersection. Therefore, the algorithms take time O(**nm**), as one would expect for a naive intersection algorithm.

7. Conclusion

The aspect representation is the first example of a *continuous* viewer-centered object representation. Using such a representation we can work with the *appearance* of an object rather than the volume of space that it fills. We have shown that this is advantageous for object recognition, and we also expect that techniques of the sort developed here will prove useful in other problems, such as computer graphics. In fact, we believe that one of the main results of this paper is a better understanding of how to work with the appearance of an object over a continuous range of viewpoints. This paper elucidates these ideas by answering questions about *aspect* itself, such as how we can calculate and represent appearance over a *range* of viewpoints, what the connections between different views of an object are and how we can represent them, and how we can search for viewpoints with a particular appearance or property.

The aspect representation has something in common with multiview object representations since any appropriate cross-section of the asp is a view of the object. However, it clearly goes beyond the multiview idea since it represents the appearance of the object 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.

Another significant advantage of the aspect representation is that it makes it easier to answer queries asking at which viewpoints a particular visual event occurs, for example, at which viewpoints a certain feature is visible. Using standard representations such queries involve computing the rotation of the object necessary for the event to occur. However, using the aspect representation, answering the query reduces to searching the asp for the given feature. Resulting algorithms may even turn out the same using both approaches except that constructing the asp solves most of the problem during preprocessing.

The aspect representation is large for highly non-convex objects because in a sense it contains every hidden-surfaces-removed image of the object. In return for the space requirements, though, the asp makes explicit a great deal of information about the object or the scene. For that reason, the asp is a natural choice for use in parallel algorithms and algorithms for problems which are difficult to compute.

Marr [1978, 1982] argues that object-centered representations are preferable over viewer-centered representations. His argument is essentially that object-centered representations are more concise, and humans can recognize objects from a wide variety of views and against a wide variety of backgrounds. However, the aspect representation is quite different from multiview 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. Second, it represents the object from all viewpoints, not just a discrete few. Furthermore, Marr's argument does not take into account the advent of plentiful memory. If we have precomputing time and plentiful space available, representations which have many different parts but the processing for each part is easier may be just what we are looking for, especially if we can design efficient parallel algorithms to process the representation.

Some open problems relating to the aspect representation are the following: what happens to aspect space and the aspect representation under perspective rather than orthographic projection? Is it possible to define an approximation to the asp which has smaller size, perhaps by constructing the asp for various levels of a hierarchical representation of an object? How does the asp generalize to objects with curved surfaces, for example, generalized cylinders? Can we represent the effect of a point light source in aspect space, so that shadows in an image are explicitly stored? Can we define the asp procedurally, as we can represent an object in E^3 procedurally, in a manner similar to constructive solid geometry?

References

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

Baumgart, B. G., "Winged edge polyhedron representation," Rep. STAN-CS-320, Stanford Artificial Intelligence Lab., Stanford U., 1974.

Castore, G., "Solid modeling, aspect graphs, and robot vision," General Motors Conf. on Solid Modeling, 1983.

Castore, G. and C. Crawford, "From solid model to robot vision," *Proc. IEEE First Int. Conf. on 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.

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

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

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.

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

Korn, M. and C. Dyer, "3D multiview object representations for model-based object recognition," to appear in *Pattern Recognition.*

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.

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.

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

Plantinga, H. and C. Dyer, "The Aspect Representation," forthcoming Technical Report, 1987.

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

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.