THE ASPECT REPRESENTATION

by

Harry Plantinga
and
Charles Dyer

Computer Sciences Technical Report #683

January 1987

The Aspect Representation

Harry Plantinga - Charles Dyer

Computer Sciences Department
University of Wisconsin - Madison
Madison, WI 53706

Abstract

In this paper we present a new continuous, viewer-centered representation for polyhedral objects called the aspect representation or asp. The representation is viewer-centered in the sense that it represents the appearance of the object rather than the volume of Euclidean space that it fills and continuous in the sense that it represents the appearance continuously from all viewpoints. We then give some properties of asps and algorithms for constructing and finding unions, intersections, and differences of asps. We also name some applications for which we expect the aspect representation to prove useful.

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

1. Introduction

Representing three-dimensional objects within a computer is important to computer-aided design, robotics, computer graphics, computer vision, and many other areas, so it is not surprising that there are many different representations in use. Major existing methods for representing objects in three dimensions can be divided into three main categories: volumetric representations, boundary representations, and swept-volume representations. Volumetric representations, such as octrees, space occupancy arrays, and constructive solid geometry represent the volume of space that the object occupies by constructing that volume as a combination of simpler volumes. Boundary representations such as winged-edge polyhedra represent the surface bounding the volume. Swept-volume representations represent the cross-sectional area of the object swept along a spine. However, all of the standard representations have something in common: they are object-centered in the sense that they represent in some manner the volume of Euclidean 3-space that the object occupies.

Object-centered representations have attractive properties. 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, there are other desirable properties that object-centered representations do not have. For example, for some applications we may want to know how an object appears to a viewer, but computing the appearance of an object to a viewer from an object-centered representation is time-consuming. For these applications it is advantageous to explicitly represent the way a three-dimensional object appears to a viewer from some or all viewpoints. Such a representation is *viewer-centered* rather than *object-centered.*

Previous viewer-centered representations for objects generally consist of a number of images of the object and are thus called multiview representations. Since they represent appearance from some discrete set of viewpoints, they make it easy to determine what the object looks like from those viewpoints. However, there are infinitely many geometrically distinct appearances of an object, and multiview representations must choose a representative few. Is there some way to represent appearance from a continuous range of viewpoints?

The answer lies in noting that the appearance (or aspect) changes *continuously* with viewpoint. We take advantage of this fact to represent the appearance of the object over ranges of viewpoints continuously. That is, we do not represent various individual two-dimensional views of the object, but rather the volume of a different space that the object occupies which includes degrees of freedom in the image plane *and* in viewpoint. In the aspect representation we will represent the volume of *aspect space* that the object occupies, where aspect space is a four-dimensional space consisting of two degrees of freedom in viewing direction (viewpoint space) and two degrees of freedom in appearance (image space).

We expect the aspect representation to have application in various problems in computer graphics and computer vision. Some of the problems for which we believe that the aspect representation will prove useful or for which we have already completed work [Plantinga & Dyer, 1986, 1987] are:

- Constructing the aspect graph
- Sequential and parallel algorithms for hidden line removal
- Answering on-line point visibility queries
- Determining 3D shape from a series of 2D views
- 3D object recognition

The term viewer-centered representation is normally used to refer to object representations which represent the appearance of an object in some manner, typically from a discrete set of viewpoints. We extend that notion to include representing the appearance or other properties of the 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.

In Section 2 of this paper we present an annotated bibliography of related work, first of other major methods currently used for representing three-dimensional objects, and then of object representations and other work related to the ideas of aspect and viewer-centered representations. In Section 3 we define the aspect representation formally and give some examples, and then we describe some properties of asps. We then discuss advantages and disadvantages of asps and address Marr's objection to viewer-centered representations [Marr, 1978, 1982]. We next present algorithms for working with asps: constructing the asp and finding the difference, intersection, and union of asps. In the last part of Section 3 we examine the size of the asp.

2. Related Work

In this section we present an annotated bibliography of related work. We first describe current techniques for representing solid objects. We then describe work related to viewpoint or aspect space and our notion of viewer-centered representations.

2.1. Current Object Representation Techniques

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 then represent the faces by the bounding line segments and the line segments by the endpoints. The representation is thus a data structure consisting of many points (the vertices of the polyhedron) and adjacency information. If the only adjacency information stored in the data structure specifies 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 the 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 [Markowski, 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. Representations with successively more adjacency information have been introduced; for example, a popular and influential representation is the *winged-edge polyhedron* representation [Baumgart, 1972], which stores all the adjacency information in space proportional to the size 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 surface curves with various types of surfaces 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, or 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 [Agin, 1977], [Brooks *et al.*, 1979], [Marr and Nishihara, 1976], [Shafer and Kanade, 1983], and [Soroka and Bajcsy, 1978].

One type of volumetric representation is the spatial occupancy representation. In this type of representation, the volume of space that the 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 one and the rest to zero ([March and Steadman, 1974], [Srihari, 1981]). A more sophisticated and 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. The space savings over the naive approach occurs in that it is only necessary to store a large number of small cubes at the surface of the 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.

2.2. 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 3-space, E^3, that the object occupies, either directly or indirectly. Is there work of a more viewer-centered flavor, that is, a flavor of representing the way the object appears from various viewpoints or making it easier to calculate the appearance of the 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.* [1980] 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. Ponce [1985] creates a hierarchical *prism tree* which also speeds intersection detection. While these techniques do not make use of viewer-centered ideas directly, their goal is to facilitate computing different views of an object or scene.

For the purposes of object recognition, the multiview representation has 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.

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 (i.e. the topology of the image of the object) from a particular viewpoint. From almost every viewpoint, the *topological* appearance of the object remains constant over small changes in viewpoint since no faces appear or disappear and in general 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* (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 amd 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* 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. Castore [1983], Castore and Crawford [1984] and Crawford [1985] encode information about *every* aspect in the aspect graph.

A problem with this sort of approach is that there can be very many different views of an object and each requires a different image. Plantinga and Dyer [1986] show that the maximum size of the aspect graph is $O(\mathbf{n}^2)$ in the convex case and $O(\mathbf{n}^4)$ in the general case for an object 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 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 connecting the different views of the object; algorithms using such a representation could as well be processing views of different

objects.

Note that our use of the term *aspect* is different from that of Koenderink and van Doorn. They define *aspect* to be the topological appearance of an object. The aspect doesn't change for most small changes in viewpoint since the topological structure of the image doesn't change. We define aspect as the geometric appearance of the object from some viewpoint rather than the topological appearance. The difference is that in our definition, the aspect changes with every change in viewpoint, no matter how small.

In other work related to aspect ideas, Werman *et al.* [1986] give an algorithm for constructing the aspect graph in the case of a convex polygon. Plantinga and Dyer [1986] give algorithms for constructing the aspect graph in both the convex and general cases under orthographic projection. 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 over changing viewpoint.

The multiple view representation idea is viewer-centered in the traditional sense of the word since the idea involves representing the way an object appears to a viewer from a number of different viewpoints. In other work of a viewer-centered flavor, 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 precomputes the regions of viewpoints at which each face obstructs the view of the feature to enable a faster answer to the queries. This work is the most closely related to *continuous* viewer-centered representation since it represents regions of viewpoint space where a visual property holds. We know of no other work which represents the appearance of an object from all viewpoints continuously.

3. The Aspect Representation

In this section we will define a particular continuous viewer-centered object representation for three-dimensional objects, namely the *aspect representation*, or *asp* for short. We will present some properties of the asp and describe how to work with asps. We will also present algorithms for constructing the aspect representation for a polyhedral object and finding the union, intersection, and difference of asps. Finally, we examine the size of the asp in various cases.

3.1. Definition and Examples

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 $\phi < 180°$.

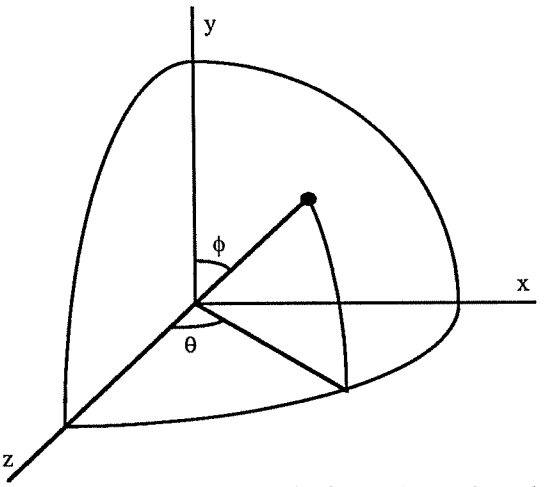

Figure 1. The Viewpoint (θ,ϕ) on the unit sphere

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:

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

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

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

$$u = x_0\cos\theta - z_0\sin\theta \quad (1)$$
$$v = x_0\sin\theta\sin\phi + y_0\cos\phi + z_0\cos\theta\sin\phi \quad (2)$$

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 $c_1 = z_1-z_0$:

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

The point equations then become

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

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.

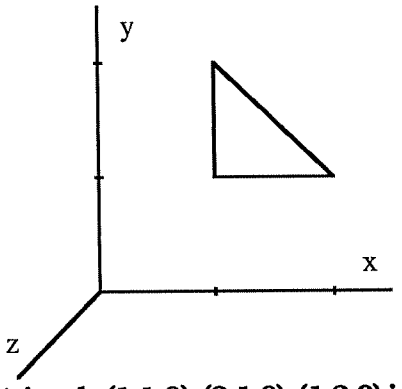

Figure 2a. The triangle (1,1,0), (2,1,0), (1,2,0) in object space.

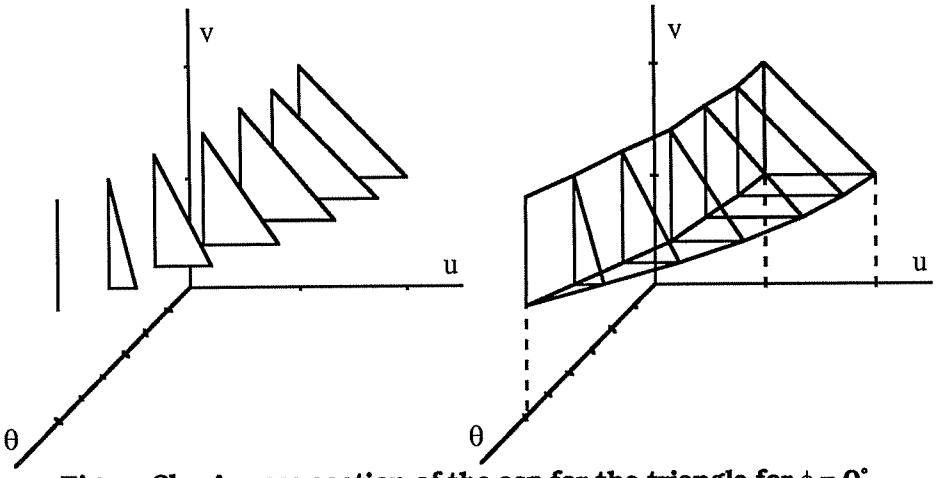

Figure 2b. A cross-section of the asp for the triangle for $\phi = 0^{\circ}$.

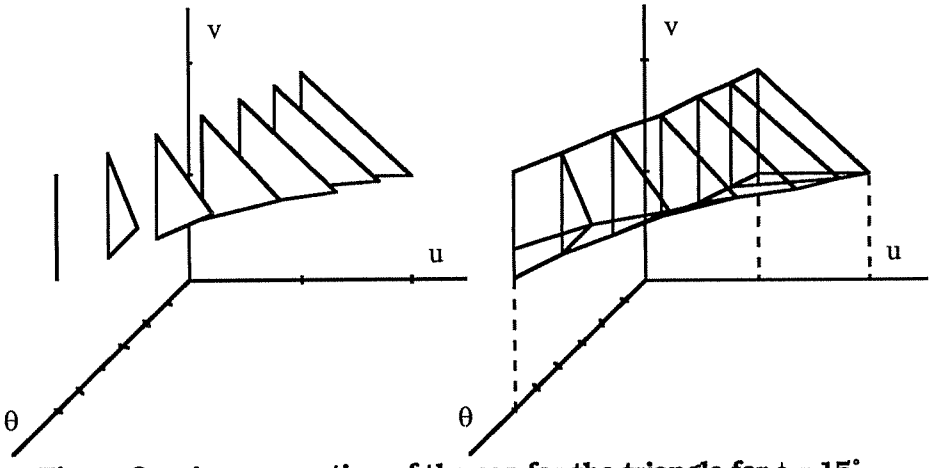

Figure 2c. A cross-section of the asp for the triangle for $\phi = 15^{\circ}$.

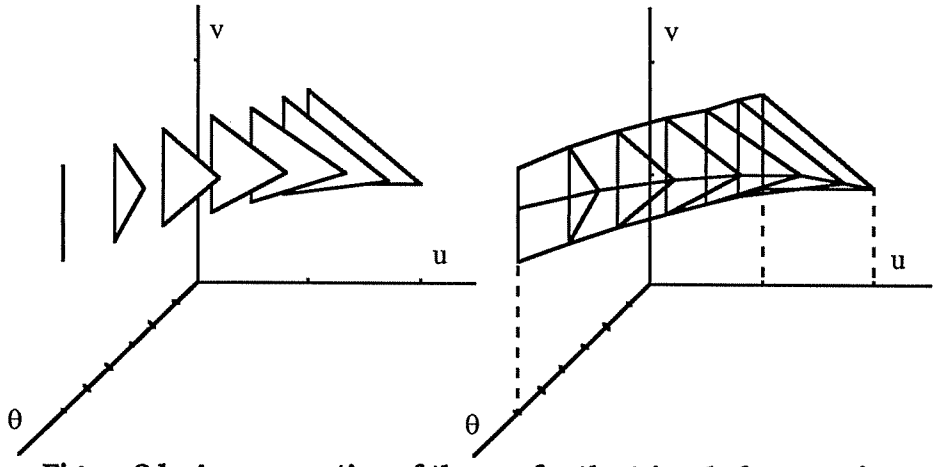

Figure 2d. A cross-section of the asp for the triangle for $\phi = 30^{\circ}$.

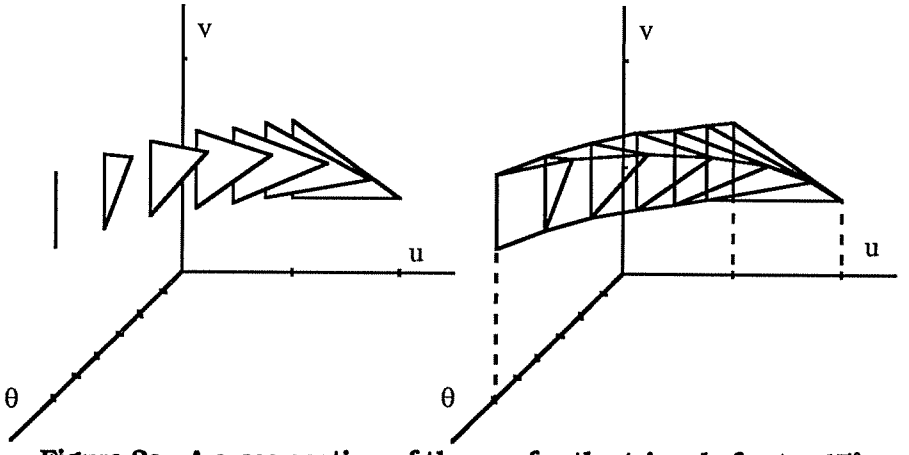

Figure 2e. A cross-section of the asp for the triangle for $\phi = 45^{\circ}$.

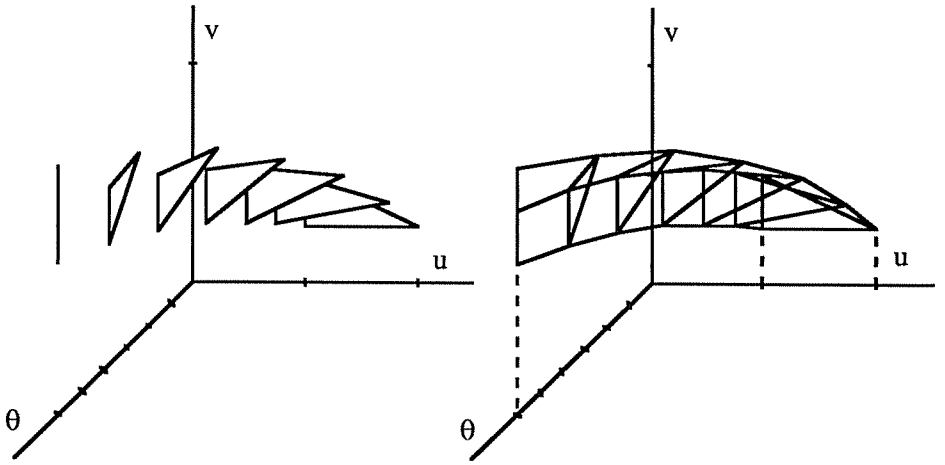

Figure 2f. A cross-section of the asp for the triangle for $\phi = 60^{\circ}$.

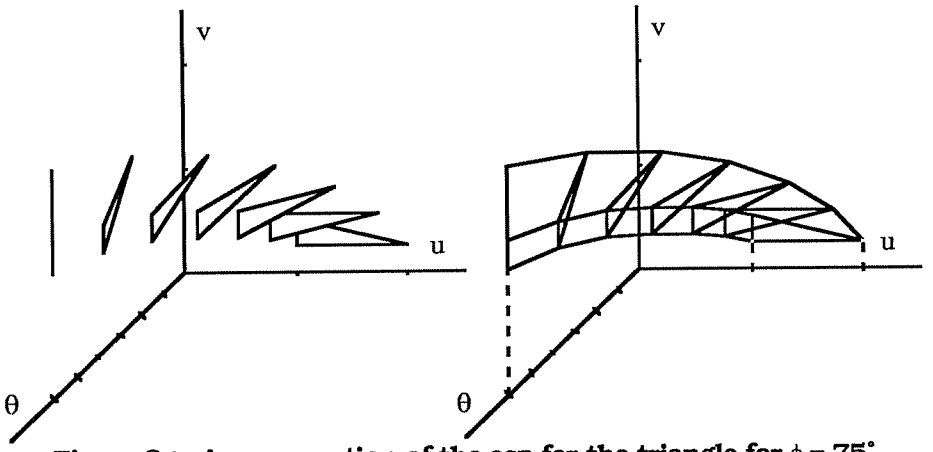

Figure 2g. A cross-section of the asp for the triangle for $\phi = 75^\circ$.

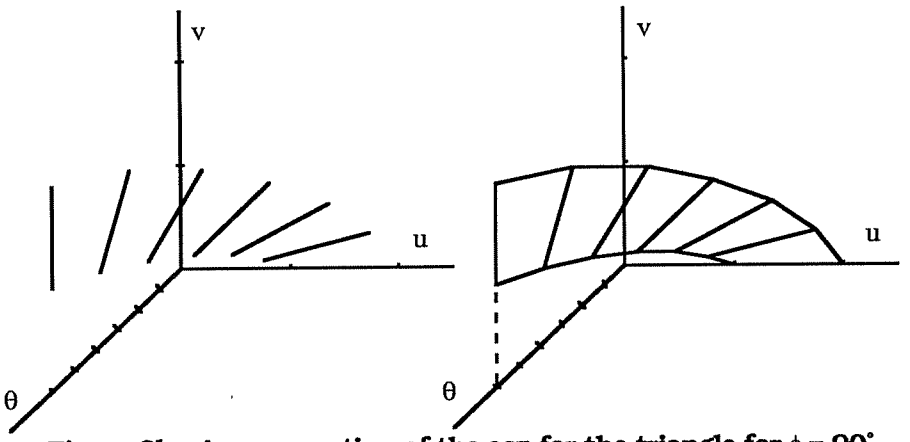

Figure 2h. A cross-section of the asp for the triangle for $\phi = 90^\circ$.

3.2. 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.

3.3. 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 $\mathbf{p}_2$.

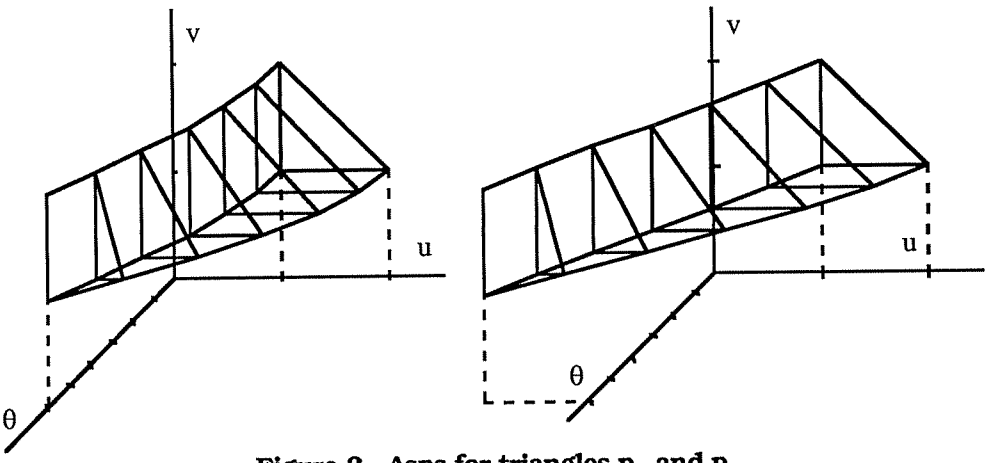

Figure 3. Asps for triangles $\mathbf{p}_1$ and $\mathbf{p}_2$.

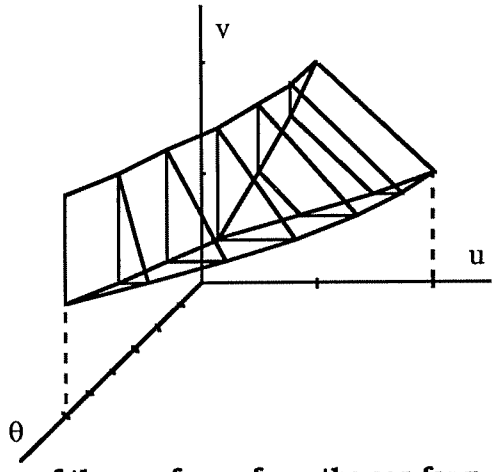

Figure 4. The subtraction of the asp for $\mathbf{p}_1$ from the asp for $\mathbf{p}_2$, which is the asp for $\mathbf{p}_1$ as obstructed by $\mathbf{p}_2$.

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 **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 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

3.4. Representing Asps

We have shown above that the surfaces bounding an asp are not hyperplanar; in this section we characterize those surfaces and show how to represent them and find intersections of pairs of them. We then present a data structure for representing asps.

3.4.1. Aspect Surfaces

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:

$$u = (x_1 + a_1 t_1) \cos\theta - (z_1 + c_1 t_1) \sin\theta \quad (6)$$

$$v = (x_1 + a_1 t_1) \sin\theta \sin\phi + (y_1 + b_1 t_1) \cos\phi + (z_1 + c_1 t_1) \cos\theta \sin\phi \quad (7)$$

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

$$u = (x_2 + a_2 t_2) \cos\theta - (z_2 + c_2 t_2) \sin\theta \quad (8)$$

$$v = (x_2 + a_2 t_2) \sin\theta \sin\phi + (y_2 + b_2 t_2) \cos\phi + (z_2 + c_2 t_2) \cos\theta \sin\phi \quad (9)$$

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

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

where

$$d1 = c2\,(y2 - y1) - b2\,(z2 - z1)$$
$$d2 = b2\,(x2 - x1) - a2\,(y2 - y1)$$
$$d3 = c2\,(x2 - x1) - a2\,(z2 - z1)$$
$$d4 = b1\,c2 - b2\,c1$$
$$d5 = a1\,b2 - a2\,b1$$
$$d6 = a1\,c2 - a2\,c1$$

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 **l** through the three object edges. We will assume that **l** is

represented parametrically in a parameter **s** and that **l** intersects $\mathbf{e}_1$ at **s**=0 and $\mathbf{e}_2$ at **s**=1. The result is a set of 9 equations in 10 unknowns. From there it is easy to solve for **l** in terms of the parameter of $\mathbf{e}_3$ using the quadratic equation. From **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.

3.4.2. A Data Structure for 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, since a four-dimensional polytope can have asymptotically more facet-ridge and ridge-edge incidences than faces. However, in the case of the aspect representation, it turns out that this data structure does have linear size because of the close connection with a polyhedron. We show this in the next section.

3.5. Size of Asps

In the case of a convex polyhedron, the asp has size $O(\mathbf{n})$ for an object with $\mathbf{n}$ 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 general four-dimensional polytope, it is not sufficient to bound the number of vertices or facets. 4-polytopes that have $\mathbf{n}$ vertices can have $O(\mathbf{n}^2)$ 2-faces. However, the asp is not a general 4-polytope, and it turns out that the size of an asp (i.e. the total number of faces) is linear in the number of vertices of the asp in the absence of degeneracy of the sort where five or more object edges appear to intersect in a single point in an image. This is because in the absence of such degeneracy, four asp edges meet in a vertex, three ridges in an edge, and so on. Thus, a bound on the number of vertices is a bound on the asymptotic size of the asp.

Each vertex arises as the apparent intersection of four object edges in an image. In the absence of degeneracy, every set of four lines can be intersected by a unique fifth line or no fifth line at all, so there is at most a single viewpoint (and its polar opposite) from which the four edges appear to intersect. Since the number of 4-tuples of lines from a set of $\mathbf{n}$ is $C(\mathbf{n},4)$, the number of vertices is bounded by $O(\mathbf{n}^4)$. In fact, the size of the asp is bounded by $O(\mathbf{n}^4)$ even with degeneracy of the sort where more than four edges appear to intersect in a single point in an image. In this case the effect is to cause some vertices to coincide. This effectively reduces the number of vertices without increasing the number of edges, ridges, and so on. Therefore with degeneracy, the size of the asp is no longer linear in the number of vertices, but the upper bound of $O(\mathbf{n}^4)$ still holds.

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 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 $\Omega(\mathbf{n}^4)$ 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 5).

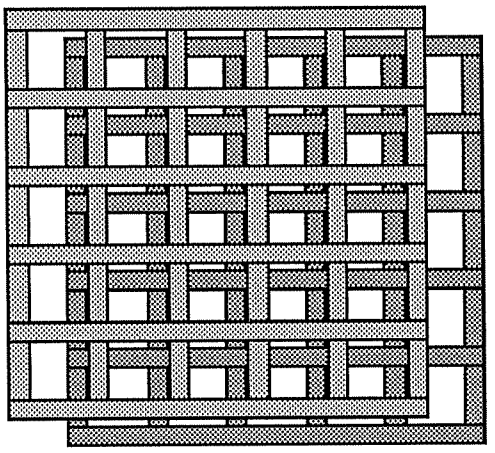

Figure 5. An example of an object for which the asp has size $O(n^4)$.

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 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.

The example in Figure 5 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 four of the strips have edges that appear to intersect in a single point. Thus, the asp for that object has $\Omega(\mathbf{n}^4)$ vertices.

In the previous section we constructed a data structure representing incidences of the asp, and we noted that it remains to be proved that the data structure has size of the order of the size of the asp. The data structure consists of pointers from faces to incident faces of one degree higher or lower dimension. For example, from edges there are pointers to incident ridges and vertices. Thus the number of pointers is proportional to the number of cell-facet, facet-ridge, ridge-edge, and edge-vertex incidences.

The number of cell-facet incidences is twice the number of facets, since each separates two cells (counting the exterior as a cell). Similarly, the number of edge-vertex incidences is twice the number of edges. Since ridges are the aspect feature corresponding to the apparent intersection of two object edges in an image and facets correspond to single object edges, the number of ridge-facet incidences is proportional to the number of times an image edge is adjacent to the intersection of two image edges. This is proportional to the number of intersections of two image edges, i.e. the number of ridges of the asp, since at most four facets can be incident upon the ridge. The remaining case, edge-ridge incidences, is similar. Edge-ridge incidences occur when two intersecting image edges are adjacent to three intersecting image edges. But this is proportional to the number of cases where three image edges intersect, i.e. the number of asp edges. Thus the number of face-face incidences is proportional to the number of faces and the data structure has linear size in the size of the asp.

3.6. 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.

3.6.1. 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.

3.6.2. 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.

3.6.3. 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.

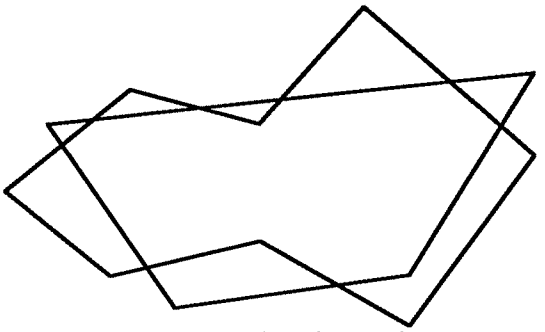

Figure 6. Overlapping polygons

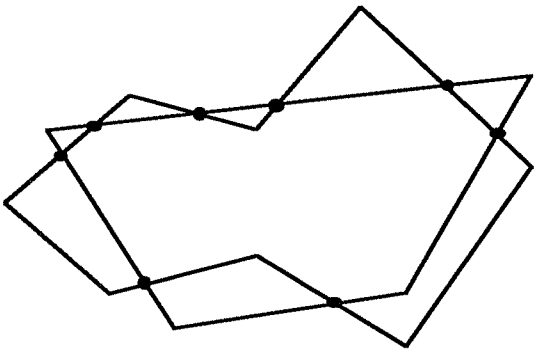

Figure 7. Finding the intersection points

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.

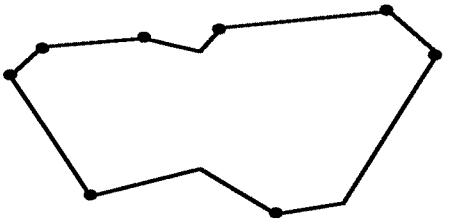

Figure 8. Cutting away the outer structure leaves the intersection

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 3.3 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.

3.6.4. 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.

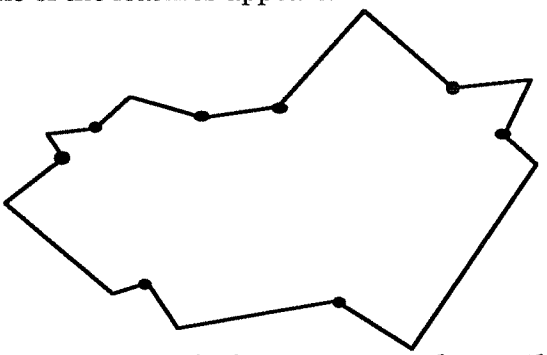

Figure 9. Cutting away the inner structure leaves the union

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.

3.6.5. 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.

3.7. Discussion

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 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 feature in question. Resulting algorithms may even turn out the same using both approaches, but the algorithms for the asp are easier to write and to understand.

The aspect representation is large, since in some sense it contains every hidden-lines-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] has argued that object-centered representations are to be preferred over viewer-centered representations. By *viewer-centered representations* he means multiple-view representations of objects. 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 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. 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.

4. Conclusion

In this paper we have introduced the asp—a new, continuous viewer-centered representation for polyhedral objects—and presented algorithms for manipulating the representation. We have also named some applications for which we expect the asp to prove useful. The aspect representation is of interest for reasons other than just the applications, however. It is also the first example of a *continuous* viewer-centered object representation, and one of the results of this work is a better understanding of the sorts of properties that viewer-centered representations have and the sorts of problems for which they are useful.

This work also elucidates the idea of *aspect* itself and answers basic questions relating to it. The following are some of the questions we have answered about aspect:

- What are the connections between different views of the same object? How can we represent them?
- How can one calculate and represent appearance over a *range* of viewpoints?
- How can one calculate and represent the range viewpoints over which a certain property holds?

The aspect representation is large for complex objects with a large number of concavities, so the objects for which it is most useful have relatively few concavities or small size. The large size of the asp and the amount of information made explicit make it a natural representation for parallel algorithms. The viewpoint-space partition representation is useful for problems that inquire about which viewpoints a particular property holds for. It has smaller size than the aspect representation, but it stores in effect one bit of information for each viewpoint (yes or no), rather than the appearance of the object.

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 through something similar to a constructive solid geometry representation?

BIBLIOGRAPHY

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

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

Baer, A., C. Eastman, and M. Henrion, "A survey of geometric modeling," Res. Rep. 66, Institute of Physical Planning, Carnegie-Mellon Univ., 1977.

Ballard, D. H. and C. M. Brown, *Computer Vision*, Prentice-Hall, Englewood Cliffs, NJ, 1982.

Barrow, H. and J. Tennenbaum, "Interpreting line drawings as three-dimensional surfaces," *Artificial Intelligence* **17**, 1981, pp. 75-116.

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

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

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

Boissonnat, J. D., "Geometric structures for three-dimensional shape representation," *ACM Transactions on Graphics* **3**,1984, pp. 266-286.

Boyse, J. W., "Data structure for a solid modeller," *NSF Workshop on the Representation of Three-Dimensional Objects*, Univ. Pennsylvania, 1979.

Brady, M., "Computational approaches to image understanding," *ACM Comput. Surveys* **14**(1), 1982, pp. 3-71.

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

Brooks, R., R. Greiner, and T. Binford, "The ACRONYM model-based vision system," in *Proc. 6th Int. Conf. on Artificial Intelligence*, vol. 1, Tokyo, Japan, 1979, pp. 105-113.

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.

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

Clark, J., "Heirarchical geometric models for visible surface algorithms," *CACM* **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. on 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. on Pattern Recognition and Image Proc.*, Dallas, TX, 1981.

Eastman, C. M., "The concise structuring of geometric data for CAD," in *Data Structures, Computer Graphics, and Pattern Recognition*, A. Klinger, K. S. Fu, and T. Kunii (Eds.), Academic Press, New York, 1977.

Fekete, G., "Generating silhouettes of polyhedra," Tech. Rep. CS-TR-1371, University of Maryland, 1983.

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.

Franklin, W. R., "Rays—new representation for polygons and polyhedra," *Computer Graphics and Image Processing* **22**, 1983, pp. 327-338.

Fuchs, H., Z. M. Kedem, and B. F. Naylor, "On visible surface generation by a priori tree structures," *ACM SIGGRAPH*, 1980.

Goad, C. A., "Special purpose automatic programming for 3-D model-based vision," *Proc. Image Understanding Workshop*, 1983, pp. 94-104.

Guzman, A., "Decomposition of a visual scene into three-dimensional bodies," Ph. D. dissertation, in *Automatic Interpretation and Classification of Images*, A. Grasseli (Ed.), Academic Press, New York, 1969.

Hanrahan, P. M., "Topological shape models," Ph. D. Thesis, University of Wisconsin-Madison, 1985.

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

Holland, S. W., L. Rossol, and W. R. 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., "Sequins and quills—representations for surface topography," in *Representation of 3-Dimensional Objects*, R. Bajcsy, Ed., Springer- Verlag, Berlin and New York, 1982.

Horn, B. K. P., "Extended Gaussian Images," *Proc. DARPA Image Understanding Workshop*, 1984, pp. 72-89.

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.

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

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

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.*

Lavin, M. A., "An application of line labeling and other scene-analysis techniques to the problem of hidden-line removal," Working Paper 66, MIT 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," in *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 Proc.* **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.

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.

Markowsky, G. and M. A. Wesley, "Fleshing out wire frames," *IBM J. Res. Devel.* **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," MIT AI Lab. Memo 377, 1976.

Newman, W. M. and R. F. Sproull, *Principles of Interactive Computer Graphics*, 2nd. ed., McGraw-Hill, New York, 1977.

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

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 Asp: a continuous, viewer-centered representation for 3D object recognition," Submitted to the International Conference on Computer Vision, London, 1987.

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

Potmesil, M., "Generation of 3D surfaces from images of pattern illuminated objects," IEEE Computer Society conf. on Pattern Recognition and Image Proc., July, 1979.

Requicha, A. A. G., "Representations of rigid solids—theory, methods, and systems," *ACM Comput. Surv.* **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 Comp. Graphics and Appl.* **2**(2), 1982, pp. 9-24.

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

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

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

Scott, R., "Graphics and prediction from models," in *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," TR CMU-CS-83-105, Carnegie-Mellon Univ., 1983.

Shapiro, L., R. Haralick, J. Moriarty, and P. Mulgaonkar, "Matching three- dimensional surfaces," *Artificial Intelligence* **17**, 1981, pp. 285-348.

Silberberg, T., L. Davis, and D. Harwood, "An iterative Hough procedure for three-dimensional object recognition," *Pattern Recognition* **17**, 1984, pp. 621-629.

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

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

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

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," *Comput. Graphics and Image Proc.* **13**, 1980, pp. 96-126.

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

Woo, T. C. and J. D. Wolter, "A constant expected time, linear storage data structure for representing 3D objects," *IEEE Trans. Systems, Man, and Cybernetics* **14**(3), pp. 510-515, 1984.