An Algorithm for Constructing the Aspect Graph

by

W. Harry Plantinga

Charles R. Dyer

Computer Sciences Technical Report #627

December 1985

An Algorithm for Constructing the Aspect Graph

W. Harry Plantinga

Charles R. Dyer

Computer Sciences Department
University of Wisconsin
Madison, WI 53706

Abstract

In this paper we consider the size and construction of aspect graphs, first in the convex case, and then in the general case. In particular, we give upper and lower bounds on the maximum size (including vertex labels) of $\Theta(\mathbf{n}^3)$ and $\Theta(\mathbf{n}^5)$ respectively, and algorithms for constructing the aspect graph which run in time $O(\mathbf{n}^4)$ and $O(\mathbf{n}^6)$. The algorithm for the general case makes use of a new object representation called the *aspect representation* or *asp*. We also show a different way to label the aspect graph in order to save a factor of $\mathbf{n}$ in the asymptotic size and construction time (at the expense of retrieval time) in both the convex and general cases, and we suggest alternatives to the aspect graph which require less space and store more information.

The partial support by the University of Wisconsin Graduate School is gratefully acknowledged.

1. Introduction

In [1], J. Koenderink and A. van Doorn consider the structure of singularities of the visual mapping for a single view of an object. After developing the notion of an "aspect," which is the topological appearance of the object from a particular viewpoint, they define an "event" as a "point on the bifurcation surface," that is, a point where a small change in the viewing angle can change the aspect. Then they define the "visual potential graph," which is a graph of regions of constant aspect which are connected by events. This graph has come to be known in computer vision as the "aspect graph."

Lately some computer vision researchers have proposed adopting the aspect graph or similar tools for various problems in computer vision, especially object recognition [2-9]. There has even been some work concerning the construction of aspect graphs [2,3,9], but to our knowledge there has been no published general algorithm for or analysis of the complexity of the graph. In this paper we address the questions of the size of the aspect graph and how to construct one, first in the case of convex polyhedral objects, and then in the general case. We then suggest an alternative to the aspect graph which requires less space and incorporates more information.

What exactly is an aspect graph? Informally, the idea is this: from any particular view of a polyhedral object, there will be some range of viewpoints from which the views of the object will be topologically equivalent. That is, the same faces will be visible and they will be in the same relationships to each other, although they may have slightly different apparent shape. With this notion we define the aspect graph: there is a vertex in the aspect graph for every such range of viewpoints, with each vertex labeled by the visible faces from that range of viewpoints. Two vertices $\mathbf{n_1}$ and $\mathbf{n_2}$ of the aspect graph are connected when some arbitrarily small change in viewing direction suffices to change the set of visible faces from the faces visible at $\mathbf{n_1}$ to those visible at $\mathbf{n_2}$ without going through any intermediate set of visible vertices (see Figure 1). Thus, given a polyhedron and its aspect graph, if one looks at the polyhedron from a viewpoint corresponding to some vertex in the aspect graph, then adjacent vertices in the aspect graph tell which faces on the polyhedron will "appear" or "disappear" as one varies the viewing angle.

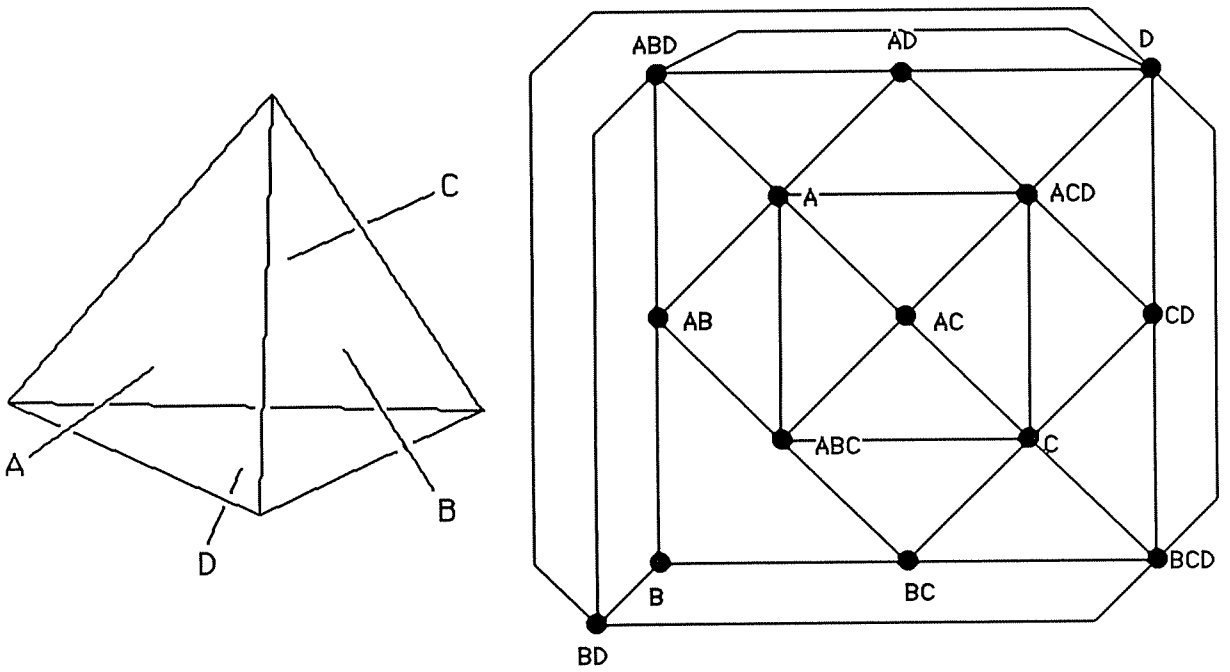

Figure 1. The Aspect Graph for a Tetrahedron.

In order to define the aspect graph more formally, we must first define "viewpoints" and "viewpoint space." In this paper we will assume parallel projection; thus we can define viewpoints as points on the Gaussian sphere centered on a polyhedron. (If one wishes to think of the viewpoints as endpoints of lines of sight (i.e. locations where one could put one's eye in order to see the appropriate view of the object), then one must think of the Gaussian sphere as being infinitely large.) Then viewpoint space is just the space of points on the Gaussian sphere. Specifically, we define the viewpoint (θ,ϕ) to be the point on the Gaussian sphere as shown in Figure 2. For example, $(\theta=0,\phi=0)$ corresponds to looking parallel to the z-axis; increasing θ corresponds to points that are increasingly "to the right" on the Gaussian sphere (with the y-axis "up"), and increasing ϕ corresponds to points on the sphere that are "higher."

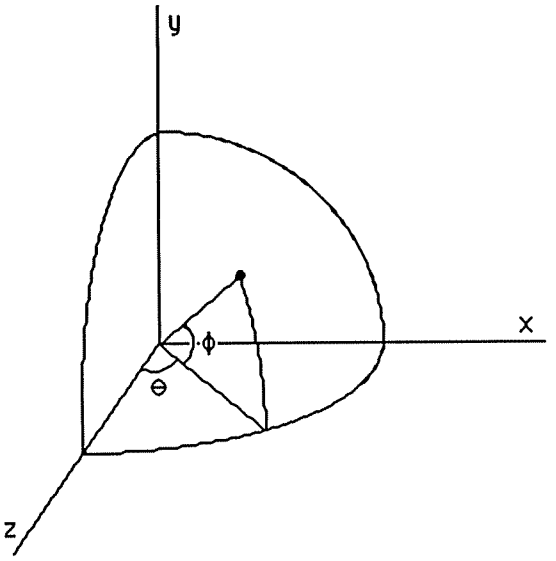

Figure 2. The Point (θ,ϕ) on the Gaussian Sphere

We define the aspect graph formally in this way: there is some subset of the set of faces of a polyhedron which are visible from any particular viewpoint. If the viewpoint changes in some direction, eventually the set of visible faces will change. Define the relation ≈ on the set of viewpoints such that $(\theta_1,\phi_1) \approx (\theta_2,\phi_2)$ for two viewpoints (θ_1,ϕ_1) and (θ_2,ϕ_2) when the set of faces visible from (θ_1,ϕ_1) is the same as the set visible from (θ_2,ϕ_2), and furthermore, one can view the object along some continuous curve of viewpoints from (θ_1,ϕ_1) to (θ_2,ϕ_2) in such a way that the set of visible faces is always the same. ≈ is an equivalence relation since it is reflexive, symmetric, and transitive.

This equivalence relation ≈ defines a partition of the viewpoint space (θ,ϕ). That is, viewpoint space is separated into connected regions such that every point (θ_1,ϕ_1) is in a region along with every other point (θ_2,ϕ_2) such that $(\theta_1,\phi_1) \approx (\theta_2,\phi_2)$. For each region of viewpoint space under this partition, there is an associated vertex in the aspect graph, which is labeled with the subset of vertices visible, and there are edges between vertices corresponding to adjacent subsets.

The observant reader may have wondered why in Figure 1 vertices A and ABC are connected, whereas vertices AB and AC are not. After all, the corresponding regions of viewpoint space seem to have the same relationship to each other in both cases: paths of viewpoints in viewpoint space connecting two viewpoints $\mathbf{v}_1$ and $\mathbf{v}_2$ in the one region and in the other all pass through a single point P (see Figure 3).

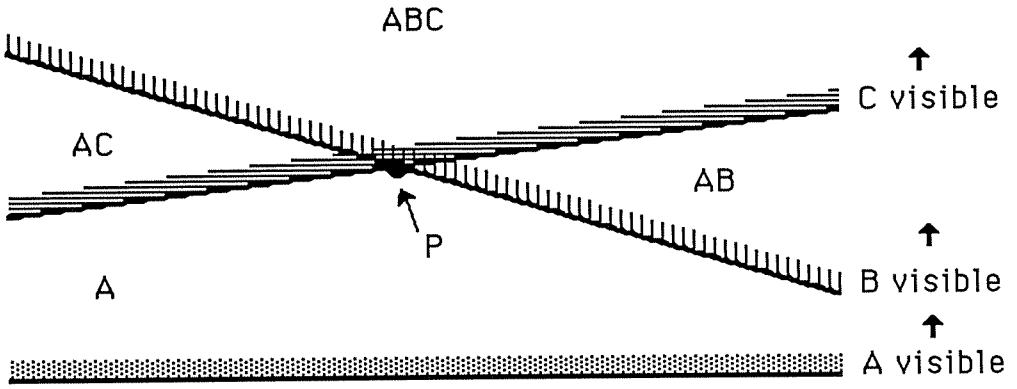

Figure 3. The Boundary between four Regions of Viewpoint Space.

It is worth pointing out here, though, that a face is not visible edge on; that is, if the line of sight lies in the plane of the face, then that face is considered not to be visible. Since the regions of viewpoint space shown in Figure 2 are regions where the same faces are visible, the boundary line is not included in the region, since on the boundary line a face has disappeared from view. Thus, the point P is in the region A but not in the regions AB, AC, or ABC.

Now it is clear why the vertices A and ABC are connected in the aspect graph while the vertices AB and AC are not: while it is possible to find a path of viewpoint from a point in region A to a point in region ABC which doesn't go through AB and AC, it is not possible to find such a path between AB and AC. If there were such a path, it would either have to go through region A, region ABC, or point P, but since point P is in region A, there is no path from AB directly to AC.

2. Convex Polyhedra

We first consider the aspect graph for convex polyhedra. This restriction greatly simplifies the problem; in the convex case, even the definition of an aspect graph is unnecessarily complex. In this case, it is sufficient to define it so that $(\theta_1,\phi_1) \approx (\theta_2,\phi_2)$ exactly when the set of faces visible from (θ_1,ϕ_1) is the same as the set visible from (θ_2,ϕ_2). However, in the non-convex case we need to be able to have separate vertices in the aspect graph for the same set of faces, since it may be that from some two different viewpoints we can see the same set of faces, yet there is no way to get between the two viewpoints without going through a viewpoint where we see some different set of faces.

2.1 A Lower Bound

In this section we prove that if a convex polyhedron has **n** faces, its aspect graph may have as many as $\Omega(\mathbf{n}^2)$ vertices and edges, and total size of $\Omega(\mathbf{n}^3)$ including vertex labels. We do this by exhibiting an example class of such polyhedra.

In Figure 4 we show a band of **m** square faces arranged in a circle. From most viewpoints, **m**/2 of the faces are visible. In Figure 5 we show two such bands arranged orthogonally around a sphere. These two bands form the essential part of our example.

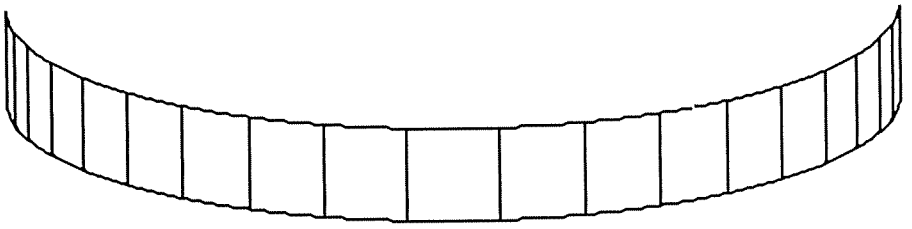

Figure 4. A Band of m Sides, of which m/2 are Visible.

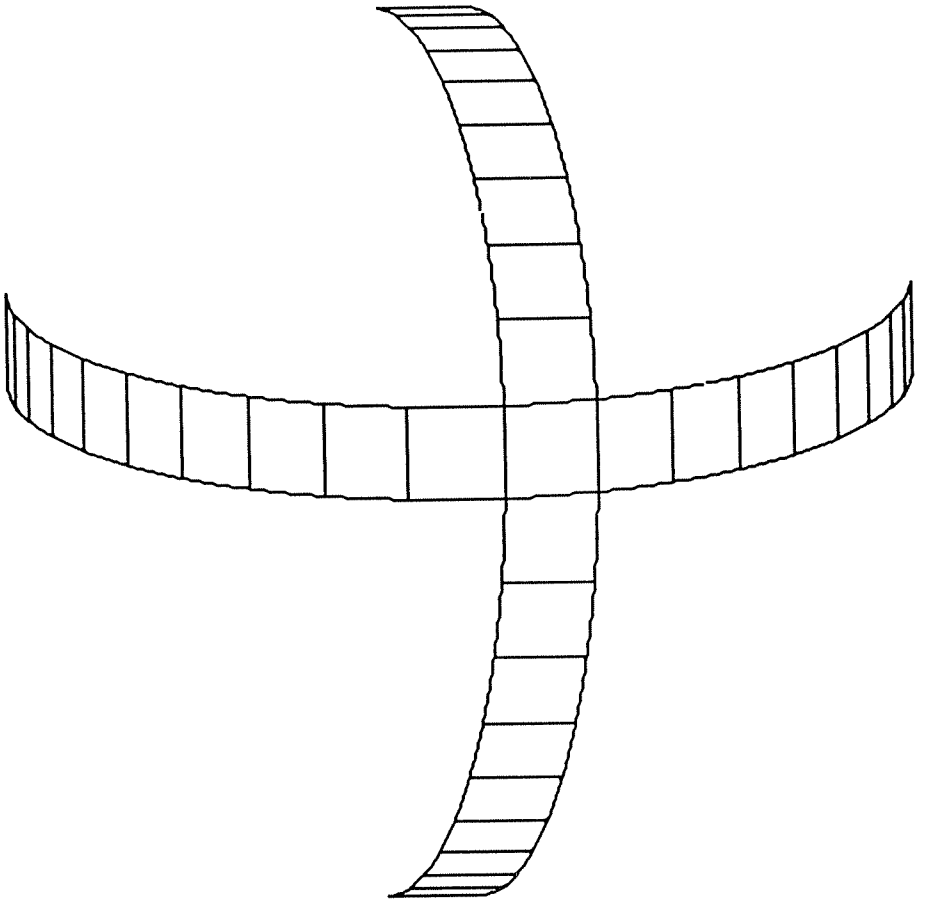

Figure 5. Two Bands around a Sphere.

Suppose that **m** is even, and consider an arbitrary contiguous subset of **m**/2 of the faces of one band; clearly there is some viewpoint from which that subset of faces is visible. (There are other viewpoints, in addition, from which only **m**/2-1 faces or no faces at all are visible, but viewpoints as described do exist.) Similarly for the other band. The important thing to notice is that for each subset of **m**/2 faces of one band, there are **m**/2 different subsets of **m**/2 faces of the other band such that there is some viewpoint from which these two particular subsets of **m**/2 faces are visible, and no others. In other words, for each set of **m**/2 vertical faces visible, we can choose a viewpoint such that one from a number of horizontal sets of faces is visible.

How many vertices must there be in the aspect graph, then, to accomodate all of these topologically distinct viewpoints? There are **m** different contiguous subsets of **m**/2 faces in each band, so there are at least $\mathbf{m}^2/2$ viewpoints from which the image of the object is topologically distinct; so there are at least $\mathbf{m}^2/2$ different vertices in the aspect graph for this object.

Furthermore, this object can be completed to a polyhedron by the addition of O(**m**) faces in the following manner: we can fill in the space between the faces between the bands with triangular faces in such a manner that the result is a convex polyhedron (see Figure 6), and we can do it with 4**m** - 16 additional faces (for **m** a multiple of 4), so that the polyhedron has a total of 6**m** - 18 faces. Thus, if we let **n** = 6**m** - 18, we have a polyhedron with **n** faces and $\Omega(\mathbf{n}^2)$ vertices in its aspect graph.

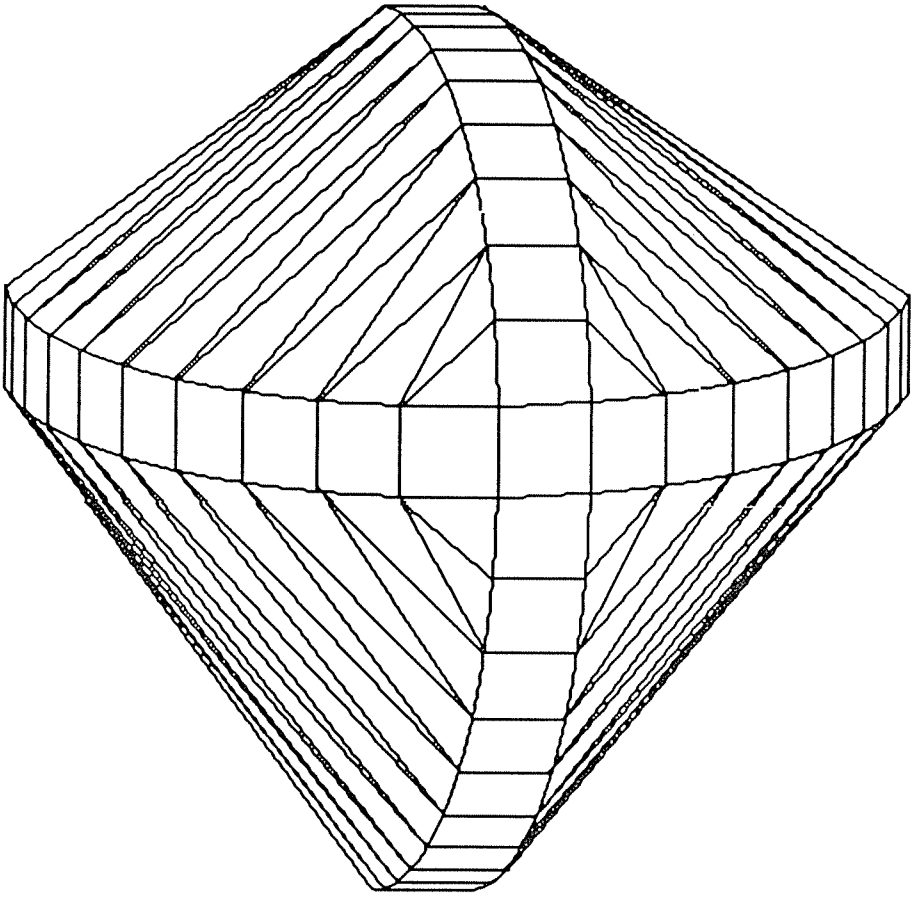

Figure 6. Forming a Convex Polyhedron.

Each node in the aspect graph has a label which names the faces visible from the corresponding viewpoints; how much do they add to its size? In this example, it is clear that each label has size $\Omega(\mathbf{n})$ to name all the faces which are visible, since there are at least **m** of them, so the total size of the aspect graph is $\Omega(\mathbf{n}^3)$.

It is possible to reduce the total storage requirement for the aspect graph by using a more efficient way to say

which faces are visible from the viewpoints corresponding to a node. A more efficient labeling system is to label one vertex with the names of the faces visible from the corresponding viewpoints and not to label the other vertices, but rather, to label the edges of the graph with the names of the faces which appear or disappear in moving between the vertices connected by the edge. To find the list of faces associated with a given vertex **v**, one would have to start with the labeled vertex and find a path from that vertex to **v**, updating the list of visible faces according to the labels of the edges in the path.

In this manner, the total storage requirement for the labels can be reduced to $O(\mathbf{n}^2)$, because along each edge at most two faces can appear or disappear at once in the nondegenerate case, that is, the case where the normals of no three faces lie on a great circle of the Gaussian sphere. If there is a degeneracy then some small change in viewing angle can result in more than two faces appearing or disappearing, but the amount of degeneracy is bounded in such a way that the total storage requirements for edge labels due to degeneracies is $O(\mathbf{n}^2)$.

We prove that the contribution to the edge label storage requirements by cases of degeneracy is less than $4\mathbf{n}^2$ face names by induction. (The first-time reader may skip the proof and continue at the next paragraph without loss of continuity.) The base case is this: suppose the polyhedron has four sides ($\mathbf{n}$=4). Then there cannot be any cases of 3-degeneracy (that is, the normals of three faces lying on a great circle) since if there were, three faces would have to be parallel to a single line; thus, the polyhedron could not be closed. The inductive step is this: suppose that the storage requirement contribution due to degeneracy for any polyhedron of $\mathbf{n}$ or fewer faces is less than $4\mathbf{n}^2$. We prove that the requirement for a polyhedron of $\mathbf{n} + 1$ sides is less that $4\mathbf{n}^2 + 2\mathbf{n} + 2$ by contradiction. Suppose we have some polyhedron with $\mathbf{n} + 1$ sides which requires more than $4\mathbf{n}^2 + 2\mathbf{n} + 2$ storage for degenerate edge labels. Put the normal points on the Gaussian sphere, and observe that removing any point results in a set of normals on the Gaussian sphere corresponding to a polyhedron with storage requirements for degenerate edge labels of less than $4\mathbf{n}^2$. Therefore the point in question was on more than $\mathbf{n} + 1$ different great circles of degeneracy, which is impossible since there are only $\mathbf{n}$ other points with which to form great circles of degeneracy. Therefore the total storage requirements for edge labels due to degeneracies is $O(\mathbf{n}^2)$.

This method of reducing the storage requirements for labels is a time-space tradeoff, however; it now takes $O(\mathbf{n}^2)$ time to find the label for a vertex on the average, since the maximum path length is $O(\mathbf{n}^2)$, and under a uniform distribution model for the start and goal vertices, the averate path length to the goal vertex from a particular start vertex is $O(\mathbf{n}^2)$. However, to find the label for a vertex adjacent to a vertex for which we have the label takes $O(\mathbf{n})$ time, equal to the time required for the unmodified aspect graph. In this paper when we say "aspect graph," we will mean the original, unmodified version; we will refer to an aspect graph modified in this manner as an "aspect graph with incremental labels."

2.2 An Upper Bound

Thus, a convex polyhedron with **n** faces can have an aspect graph with as many as $\Omega(\mathbf{n}^2)$ vertices; we new show that $O(\mathbf{n}^2)$ is also an upper bound on the size of the aspect graph (not including labels). Since each label has maximum size $O(\mathbf{n})$, we will have shown an upper bound of $O(\mathbf{n}^3)$ on the size of the whole aspect graph.

To see that the number of vertices in the graph is $O(\mathbf{n}^2)$, consider the following argument: each face on the polyhedron is visible from exactly half of the viewpoints on the Gaussian sphere; in fact, there is a great circle which separates the viewpoints from which the face is visible or invisible. Now two great circles intersect in exactly two points (or coincide), so there are fewer than $\mathbf{n}^2$ intersection points among the **n** great circles, taking coincident great circles to have no points of intersection since they correspond to parallel faces on different sides of the polyhedron. These great circles tesselate the viewing sphere in such a way that the regions between the arcs correspond to regions of viewpoints from which the same set of faces is visible, so in fact the graph consisting of vertices which are intersection points and arcs of great circles connecting them, which we will call the intersection graph, is the dual of the aspect graph. Since the intersection graph is planar and has $O(\mathbf{n}^2)$ vertices, we get that it has $O(\mathbf{n}^2)$ faces by Euler's formula, so that the aspect graph has $O(\mathbf{n}^2)$ vertices.

We claim that the aspect graph is planar; thus it must also have $O(\mathbf{n}^2)$ edges, so the total size of the aspect graph not including labels is $O(\mathbf{n}^2)$. We prove that it is planar by contradiction: suppose that the aspect graph were not planar; that is, there are four regions, A, B, C, and D, with A and B connected and C and D connected and the edges crossing. Since A, B, C, and D are disjoint, there is some point P (which is the intersection of two curves of viewpoints, one from A to B and one from C to D) which is in A and is a limit point of B (or vice versa) and which is in C and is a limit point of D (or vice versa). If there is such a point, it is in two of the regions, say A and C. But that is a contradiction: it implies that the set of faces visible from P is different from the set of faces visible from P.

2.3 An Algorithm

In this section, we give an algorithm for constructing the aspect graph which runs in optimal time for worst-case graphs. The algorithm uses the same ideas that are used in the proof of the last section. Essentially, the algorithm involves finding the way the **n** great circles for the faces tesselate viewpoint space, putting a vertex in the aspect graph for each resulting region, and connecting vertices corresponding to adjacent regions.

Specifically, the algorithm is the following: calculate the **n** great circles, and for each, calculate the 2**n**-2 intersection points (which are not necessarily unique among the great circles). The next step is to construct the graph which has these intersection points as vertices and arcs of great circles connecting intersection points as edges; we do this in the following manner: Sort all of the intersection points by their coordinates so that locating identical intersection points can be done quickly, and then for each great circle, construct a graph which consists of a vertex for each intersection point and edges between adjacent intersection points, i.e., the intersection graph. Now merge together the vertices for intersection points with the same coordinates in all of these graphs. The resulting graph is the dual of the aspect graph, so write out a graph which has a vertex for every "region" in the original graph, and edges connecting vertices corresponding to adjacent regions (see Figure 7). We must next label the vertices of this graph; we do this by checking for some viewpoint in each region which faces are visible, and labeling the vertex accordingly.

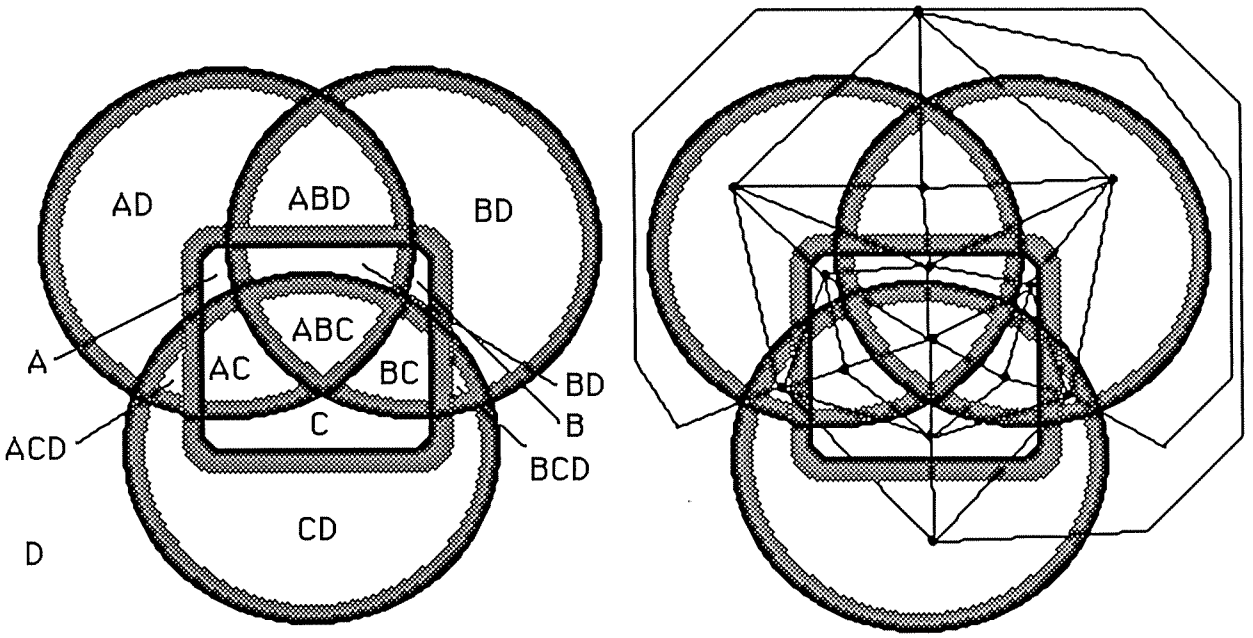

Figure 7. Regions of Viewpoint Space for the Tetrahedron and Constructing the Aspect Graph from the Intersection Graph (Compare Figure 1).

The graph that results after the merge step is the graph consisting of vertices which are points of intersection of great circles and edges connecting vertices which are adjacent along some great circle. The algorithm results in the dual of this graph, that is, the graph which has a vertex for each region in the original graph and which has edges connecting vertices for adjacent regions. Since the regions for which we have made vertices are exactly the intersections of regions where faces are visible, each region is a region where some set of faces is visible and the rest are invisible, that is, each vertex in the constructed graph is a vertex of the aspect graph. And since we connected vertices for adjacent regions, the graph resulting from this algorithm is in fact the aspect graph.

Now let us consider the runtime of this algorithm. Calculating the $O(\mathbf{n}^2)$ intersections can be done in $O(\mathbf{n}^2)$ time, since calculating the intersection of two great circles can be done in constant time, and sorting the intersection points by coordinate can be done in $O(\mathbf{n}^2 \log \mathbf{n})$ time. Finding each other vertex which should be merged with a given vertex takes $O(\mathbf{v} \log \mathbf{n})$ time, where **v** is the number of vertices to be merged, and merging them takes time linear in the number of edges merged. Since each edge is moved at most once, the whole merge step can be done in $O(\mathbf{n}^2 \log \mathbf{n})$ time. Constructing the dual of this graph can be done in time linear in the size of the graph, so the whole construction to this point can be done in $O(\mathbf{n}^2 \log \mathbf{n})$ time. Determining which faces are visible from each vertex and labeling the vertex can be done in time $O(\mathbf{n}^2)$ per vertex, so the labeling step can be done in time $O(\mathbf{n}^4)$.

3. Non-convex Polyhedra

Constructing aspect graphs in the non-convex case is more difficult: some examples of the difficulties that arise are that a face may be visible in several unconnected regions; the same set of faces may require several vertices in the aspect graph; and a particular face which is visible from some viewpoint may nevertheless not have any visible vertices or edges. The size of the aspect graph for a non-convex polyhedron can also be much larger. Since a particular face of the polyhedron can be visible in several unconnected regions of viewpoint space, there can be several different nodes in the aspect graph for a particular subset of faces.

3.1. A Lower Bound

How large can the aspect graph be in the non-convex case? In this section we show a class of polyhedra for which the aspect graph has size $\Omega(\mathbf{n}^4)$ without labels, and size $\Omega(\mathbf{n}^5)$ with them. In Figure 8 we see two polyhedra in the form of grids, one large and one small. Let each grid have **n** x **n** strips; and let the smaller grid be behind the larger one by some distance. The strips in the larger grid are wider than the whole width of the smaller grid. This whole setup is not polyhedral, but it is easy to add some more faces so that the setup *is* polyhedral, has size $O(\mathbf{n})$, and has geometry essentially equivalent to this setup.

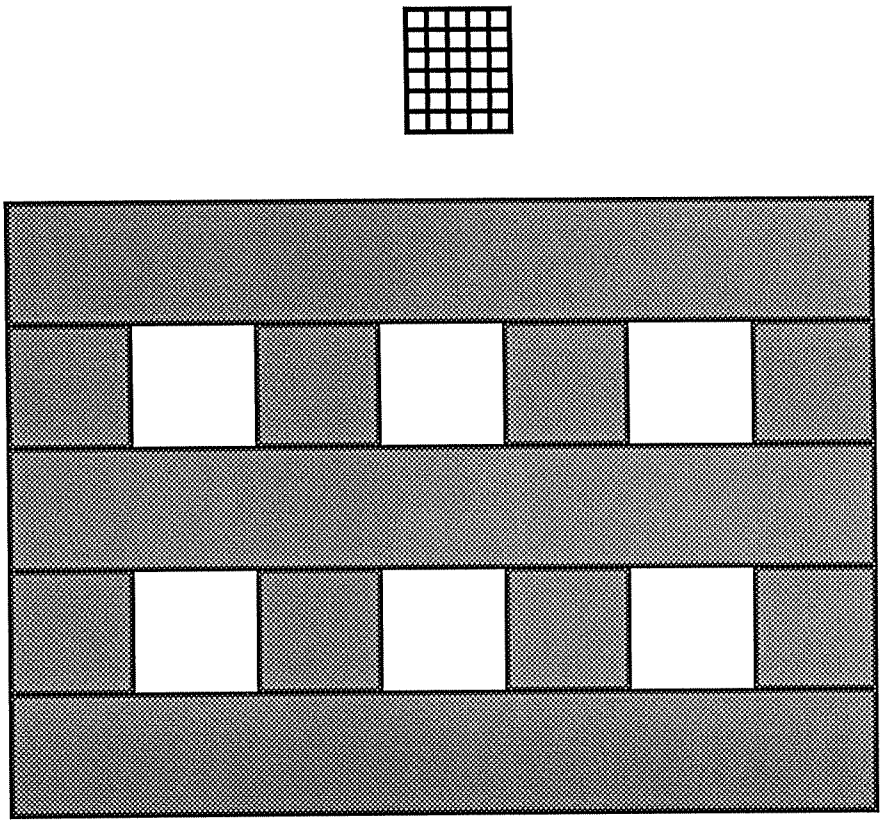

Figure 8. Two Grids which Together Have an Aspect Graph of Size $O(n^4)$.

Let us only consider viewpoints in front of the plane of the larger grid. Every (frontal) face of the larger grid is visible from all viewpoints, but the smaller grid is partially or completely invisible from some viewpoints (see Figure 9).

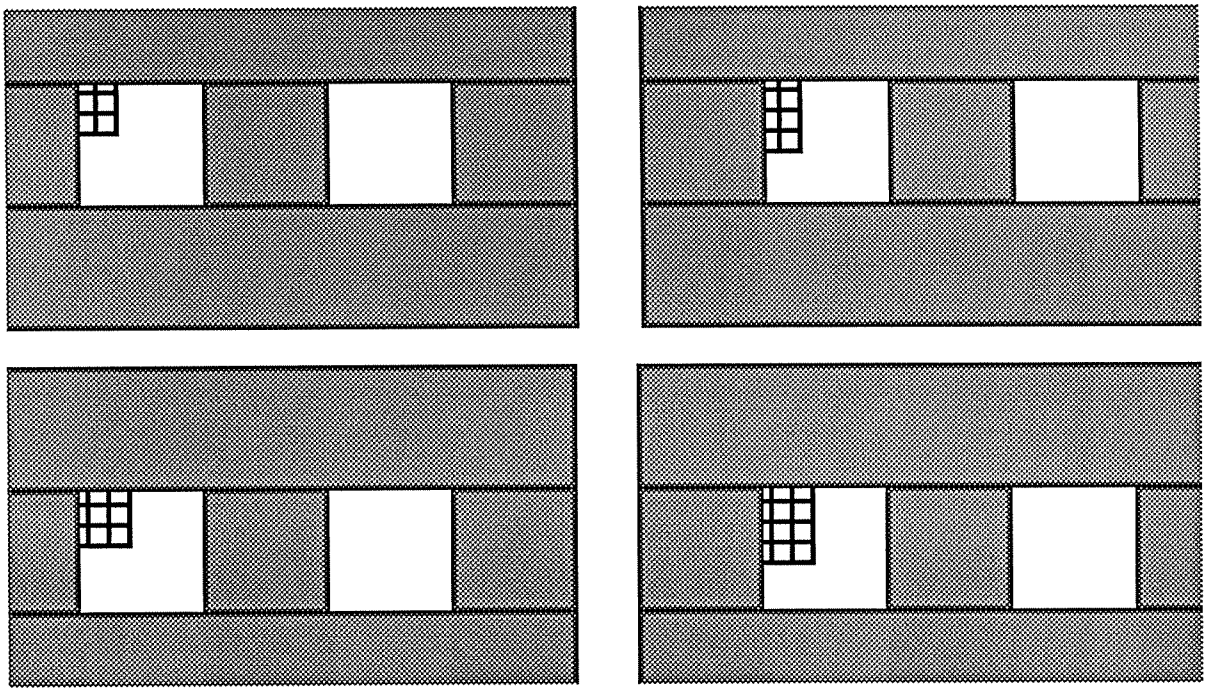

Figure 9. Different Views of the Smaller Grid

Recall that two viewpoints require different nodes in the aspect graph when the set of visible faces is different for the two viewpoints and when the set of visible faces is the same but there is no path of viewpoints from one viewpoint to the other such that the same set of faces is visible from every viewpoint on that path. We claim that these two conditions require that there be $\Omega(\mathbf{n}^2)$ different vertices in the aspect graph for every "hole" in the larger grid, and that no node is shared by two different holes, so since there are $\Omega(\mathbf{n}^2)$ holes in the larger **n** x **n** grid, there are $\Omega(\mathbf{n}^4)$ vertices in the aspect graph for this polyhedron.

There must be $\Omega(\mathbf{n}^2)$ different vertices in the aspect graph for every hole in the larger grid because there are views of the smaller grid through that hole such that any subset of **j** bottom-most of the **n** horizontal faces and any subset of **k** right-most of the **n** vertical faces of the smaller grid are visible. (Figure 9 shows **j**=3,**k**=2; **j**=4,**k**=2; **j**=3,**k**=3; and **j**=4,**k**=3 examples.) So since a different set of faces is visible for any **j** and **k** from 1 to **n**, there must be $\Omega(\mathbf{n}^2)$ different vertices in the aspect graph for views of the smaller grid through that particular hole in the larger grid. Furthermore, no node in the aspect graph corresponds to views of the smaller grid through two different holes, because there can be no path from the one viewpoint to the other with the same faces visible along the whole path: the smaller grid must "pass behind" one of the faces in the larger grid, and since the faces in the larger grid are wider than the whole smaller grid, from some viewpoint it must disappear completely. And since the larger grid is always visible and has q(**n**) faces, the vertex labels have size q(**n**). Therefore, the size of the whole aspect graph is $\Omega(\mathbf{n}^5)$.

3.2. An Algorithm

The algorithm for the aspect graph in the case of convex polyhedra was fairly straightforward; it made use of a representation for the region of viewpoint space in which each face is visible. In fact, it involved using these regions to construct a tesselation of viewpoint space which turned out to be the dual of the aspect graph. The problem with this approach in the case of non-convex polyhedra is that one face on the surface of the polyhedron can obstruct the view of another face, so that the region in which a face is visible is not in general the region bounded by a great circle. However, perhaps we can determine the region of viewpoint space in which each face is visible; we could then draw all of these regions on the Gaussian sphere, and the resulting tesselation would be the dual of the aspect graph. But how can we calculate the region(s) of viewpoint space in which a face is visible?

A first approach might be to start with the half-sphere in which some face **f** would be visible if the polyhedron were convex, and for each other face in the polyhedron to subtract the region of viewpoint space in which that face obstructs the view of **f**. The problem with this approach is that there may be some viewpoint from which **f** is not visible, and yet each other face only partially obstructs the view of **f**. That is, there is no face which totally obstructs the view of **f**, and yet it is not visible.

We might try modifying this approach a bit; perhaps for each face **f** we should determine the region of viewpoint space where each other face *partially* obstructs the view of **f**. But there is a problem with this approach, too: for any given viewpoint, we may know that *all* of the faces partially obstruct its view and still not know whether some part of **f** is visible or not.

What we need is a way to represent what part of the view of **f** is obstructed by each face, at each viewing angle. Then for any viewpoint we could determine whether some part of **f** is visible. In fact, we need something stronger than this: it is not sufficient to be able to determine that part of **f** which is obstructed for any particular viewpoint; if we want our algorithm to terminate we can't check an infinite number of viewpoints. It would be nice to be able to determine that part of **f** which is visible in a viewpoint- independent way; that is, to be be able to determine for *every* viewpoint at once the part of **f** which is visible from that viewpoint.

3.2.1. The Asp

In order to work more easily with such seemingly elusive notions as how one face obstructs the view of another from *every* viewing angle simultaneously, we introduce a new representation for polyhedral objects. Existing object representations--direct representations of polyhedra, octrees, volumetric representations, constructive solid geometry, etc.--all represent the volume of Euclidean space which the object occupies and are therefore *object-centered* representations. We introduce a *viewer-centered* representation: it represents the volume of *viewer-space* that the object occupies, where viewer-space is defined to be (θ,ϕ,x',y')--that is, the object's (x',y') projection in the viewing plane for any viewpoint (θ,ϕ). (Viewer-space should not be confused with viewpoint space.) We call this new representation the *aspect representation* for an object, or the *asp* for short.

Since the aspect representation for an object represents the volume of viewer-space, (θ,ϕ,x',y'), that the object occupies, the set subtraction of the asp for one polygon from the asp for another polygon represents the volume of viewer-space in which a point of the second polygon is visible. Thus, subtracting the asp for every other face from the asp for one particular face **f** and taking the projection of that asp onto (θ,ϕ) results in the region(s) of (θ,ϕ) in which **f** is visible.

3.2.2. Properties of Asps

We represent an asp by representing the volume of viewer-space that the asp occupies; specifically, by representing the 3-manifolds that bound the 4-manifold. Unfortunately, the "faces" aren't in general planar, so the object is not a polytope. However, every (x',y')-cross section of the object is made up of polygons, since the appearance of a polyhedron from any viewpoint is polygonal. Also, every arc on the projection of an asp onto the viewpoint sphere (θ,ϕ) is the arc of a great circle, since if we think of an arbitrarily large sphere around the polyhedron, and consider each point on the sphere as a point on a line of sight, and if we imagine the endpoint of some edge as a point light source, the shadow cast by any other line segment is arbitrarily close to an arc of a great circle. Thus, any (θ,ϕ)-cross section of the asp for a polyhedron is spherical polygonal.

Thus, while the asp is not a polytope, we will speak of it as if it were. Specifically, we will refer to the 3-manifolds that bound the asp as "facets," the 2-manifolds that bound the facets as "ridges," and the 1-manifolds as "edges." We also claim that we can work with asps in a manner very similar to that in which we work with polytopes; for example we can determine whether two asps intersect by determining whether some facet of one intersects some ridge of the other, and if not, whether one is completely inside the other. In particular, we claim that it is possible to construct the intersection of two asps or the set subtraction of one from the other in time $O(\mathbf{nm})$, where **n** and **m** are the sizes of the two asps. We will examine these claims in more detail in a future paper.

3.2.3. Using Asps to Find the Aspect Graph

Briefly, the algorithm for constructing the aspect graph for a non-convex polyhedron is this: for every face **f** in the polyhedron, do the following steps. First, make a list of all the faces and partial faces that lie in front of the plane of **f** by taking the intersection of each face with the plane of **f**. Then construct the asp for **f** and subtract (set subtraction) the asp for all the other polygons on the list of partial faces which lie in front of **f**. What remains is the asp for **f** as obstructed by all of the faces in front of **f**. Take the (θ,ϕ)-shadow of this asp, that is, the outline of the projection of the asp onto (θ,ϕ). The regions of (θ,ϕ) which result are the regions of viewpoint space where that face is visible.

Next, construct the visibility region of (θ,ϕ) for each face in the polyhedron similarly. Find the resulting partition of (θ,ϕ), and for each region of the partition, make a vertex in the aspect graph. Connect vertices corresponding to adjacent partitions. Now label the vertices in the aspect graph by checking for each vertex which of the face visibility regions it intersects.

This algorithm can be expressed more succinctly: construct the asp for the polyhedron. Find the (θ,ϕ)-shadow. Make a vertex in the aspect graph for each region in the (θ,ϕ)-projection, and connect vertices corresponding to adjacent regions. For each region, determine which faces are visible from some point inside the region by checking the asp, and label it accordingly.

3.3. An Upper Bound

Earlier we saw that the size of the aspect graph for a non-convex polyhedron was bounded from below by $\Omega(\mathbf{n}^4)$; what is the upper bound? In fact, the upper bound is also $O(\mathbf{n}^4)$. One can see this by considering how the aspect graph is constructed: several arcs of great circles are drawn on the sphere of viewpoints; the intersection graph of these arcs is constructed, and the aspect graph is the dual of the intersection graph. If we knew how many of these arcs on the sphere there could be, we would have an immediate bound on the size of the aspect graph.

But the arcs on the sphere are exactly the boundaries of regions of viewpoints where the same set of faces is visible. Therefore, the boundaries themselves consist of viewpoints where a face can "appear" or "disappear" with an

arbitrarily small change in viewpoint. Now a face can appear or disappear upon an arbitrarily small change in viewpoint only if one vertex of the face is exactly in line with the edge of another face along the line of sight, or if one edge of the face is behind, parallel to, and exactly in line with the edge of another face along the line of sight. But there are $O(\mathbf{n})$ edges and $O(\mathbf{n})$ vertices in the polyhedron, so there are $O(\mathbf{n}^2)$ great circles on which lie all of the arcs drawn on the sphere of viewpoints.

Now any two great circles coincide or intersect in exactly two points, and if they coincide they do not add any vertices to the intersection graph, so the intersection graph has $O(\mathbf{n}^4)$ vertices. The intersection graph is planar, and the aspect graph is the dual of the intersection graph. Earlier we showed that the aspect graph is also planar, so the aspect graph has size $O(\mathbf{n}^4)$, and $O(\mathbf{n}^5)$ with labels.

3.4. Complexity of the Algorithm

What is the complexity of this algorithm for constructing the aspect graph for non-convex polyhedra? The algorithm involves taking the asp for each face and subtracting the asps for faces in front of it. Now these faces may not be convex, but if they are not, we could triangulate them, and we would end up with a number of convex triangular faces, some of which happened to be co-planar. Since each face has an edge of the original polyhedron, though, the total number of faces must be linear in the original size of the polyhedron. So let us assume that the faces of the polyhedron are triangular; if the aren't, triangulate them, resulting in a linear number of faces.

The algorithm involves taking the asp for a face and subtracting the asps for the other faces. The asp for the face in question may end up in a number of pieces of varying size; we would like to know the maximum size of the asp for a face. But that size is bounded by the order of the number of vertices in the asp for that face, so the problem reduces to finding the maximum number of vertices in the asp for the face. But a vertex can occur only at the intersection of four 3-hyperplanes on which lie four facets, and these 3-hyperplanes corespond to lines in (x',y') which edges of the polyhedron lie on, swept into (θ,ϕ). There are $O(\mathbf{n})$ such 3-hyperplanes, and one of them must come from an edge of the face for which we are constructing the asp, so there are $O(\mathbf{n}^3)$ such subsets of size four. Therefore the asp for a face has maximum size $O(\mathbf{n}^3)$.

Thus, our algorithm involves subtracting an asp for a face with three or four sides from an asp of size $O(\mathbf{n}^3)$--an operation taking $O(\mathbf{n}^3)$ time. All $O(\mathbf{n})$ such operations together then take time $O(\mathbf{n}^4)$. Finding the asp for all $O(\mathbf{n})$ faces then takes time $O(\mathbf{n}^5)$. Projecting these asps onto viewpoint space takes linear time, and constructing the intersection graph can be done in time $O(\mathbf{n}^4 \log \mathbf{n})$, since it has size $O(\mathbf{n}^4)$ (in the same manner as the convex

case). Constructing the aspect graph can be done in time linear in the size of the intersection graph, which is $O(n^4)$, so the whole algorithm for constructing the aspect graph (without labels) takes time $O(n^5)$. Labeling the aspect graph can be done in time $O(n^2)$ per vertex for a total time of $O(n^6)$, so the whole algorithm takes time $O(n^6)$.

5. Conclusion

In this paper we have shown upper and lower bounds for the aspect graph of convex and non-convex polyhedra. Aspect graphs for convex polyhedra are big, and for non-convex polyhedra they're bigger still, but for those cases in which they still turn out to be a useful tool, we have also presented a worst-case optimal algorithm for constructing them.

While this paper examined the aspect graph, one of the important results is that the aspect graph is so large that it will not be a practical tool in most situations. However, two of the data structures which were used in constructing the aspect graph in the non-convex case have smaller size than the aspect graph and contain more information. One of these is the intersection graph which, together with the locations of the vertices and labels on the edges indicating which face appears or disappears, requires $O(n^4)$ space. This is a tesselation of viewpoint space, so not only does it contain all of the information of the aspect graph, but also it contains information about *exactly which* viewpoints are associated with each node in the aspect graph. Furthermore, using more storage it is possible to build a data structure so that it is possible to determine whether a particular point or face is visible from a particular viewpoint in logarithmic time.

The other interesting data structure which was introduced is the asp, which also has maximum size $O(n^4)$. An example of the interesting properties of the asp for a polyhedron is that the asp has hidden line information built into it: to find what the polyhedron looks like from some viewpoint (θ,ϕ) with hidden lines removed, it is only necessary to take the (θ,ϕ)-projection of the asp. We will consider the asp in greater depth in future work.

Throughout this paper we have assumed parallel projection. In the case of perspective projection, the aspect graph will have even more vertices since some faces will be visible from some viewing direction only when the viewer is far enough away from the object. That is, along some viewing direction but at a great enough distance, the faces visible will be the same as in the case of parallel projection, but at lesser distances from the object fewer and fewer faces will be visible. We will also consider this case in a future paper.

REFERENCES

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

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

[3] C. Crawford, "Aspect graphs and robot vision," IEEE Conf. on Computer Vision and Pattern Recognition, 1985, 382-384.

[4] M. Korn and C. Dyer, "3D multiview object representations for model-based object recognition," Technical Report 602, Dept. of Computer Sciences, Univ. of Wisconsin, Madison, 1985.

[5] C. Thorpe and S. Shafer, "Topological correspondence in line drawings of multiple views of objects," Technical Report CMU-CS-83-113, Dept. of Comp. Sci., Carnegie-Mellon University, 1983.

[6] R. Scott, "Graphics and prediction from models," in Proc. Image Understanding Workshop, 1984, 98-106.

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

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

[9] J. Galtieri, "Automatic construction of aspect graphs in prolog," Working Report, National Bureau of Standards, Sept. 1984.