Computing the Aspect Graph for Line Drawings of Polyhedral Objects

ZIV GIGUS, STUDENT MEMBER, IEEE, AND JITENDRA MALIK, MEMBER, IEEE

Abstract—**We have developed an algorithm for computing the aspect graph for polyhedral objects. The aspect graph is a representation of 3-D objects by a set of 2-D views. The set of viewpoints on the Gaussian sphere is partitioned into regions such that in each region the qualitative structure of the line drawing remains the same. At the boundaries between adjacent regions are the accidental viewpoints where the structure of the line drawing changes—a visual event occurs. We show that for polyhedral objects there are two fundamental visual events: 1) the projections of an edge and a vertex coincide, and 2) the projections of three nonadjacent edges intersect at a point. The geometry of the object is reflected in the locus of the accidental viewpoints—the boundaries of the partition. The algorithm computes the partition together with a representative view for each region of the partition. In the course of presenting the algorithm, we provide a full catalog of the changes that occur in the views during each fundamental event.**

Index Terms—**Aspect graphs, line drawing interpretation, model based vision, object recognition, polyhedral objects.**

I. Introduction

RECOGNIZING three-dimensional objects in two-dimensional images is a major area of research in computer vision. Surveys of current approaches for solving the three-dimensional object recognition problem may be found in [3] and [4]. One of the main difficulties in solving this problem is matching 2-D image information to the 3-D object representation. The multiple 2-D views approach has been suggested for reducing this problem to a 2-D to 2-D matching problem. In this approach a finite set of two-dimensional views of the object from different viewpoints is computed and the image is matched against this set. One would like the set of views to be as small as possible and yet be representative of the infinite number of possible views.

Koenderink and van Doorn [15] introduced the idea of using the *aspect graph* of topologically distinct views of an object to represent its shape. Informally, at each vertex of the aspect graph there is a view—*an aspect*—that is representative of the projections of the object from a connected set of viewpoints from which the object appears qualitatively similar. Two aspects are adjacent in the graph if the corresponding sets of viewpoints are adjacent. A *visual event* is said to occur when the view changes as the observer moves between adjacent sets.

We present an algorithm for constructing the aspect graph for polyhedral objects under orthographic projection. In Section II we present preliminaries that are necessary for the precise definition of the notion of topologically distinct views. This definition is followed by the definition of the *viewing data of an object*, which is an extension of the aspect graph. Section III is a review of previous work on the multiple 2-D views approach to object representation. In Section IV we describe the visual events for polyhedral objects and the relation between the structure of the object and the locus of accidental viewpoints. Section V is an overview of the algorithm. A detailed description of the main step of the algorithm and a catalog of the visual events for polyhedral objects are provided in Section VI. Complexity analysis is presented in Section VII. The conclusions in Section VIII include a comparison between this algorithm and the one due to Plantinga and Dyer.

II. Preliminaries

A. The Labeled Image Structure Graph

In the projection of a polyhedral object, every line in the image is the projection of an edge of the object. Every edge is classified as convex or concave according to the dihedral angle, inside the object, between the faces that share the edge. A convex edge may also be classified as an *occluding* edge if, from the given viewpoint, the faces that meet along the edge are on the same side of the edge.

In a *labeled line drawing* each line is labeled according to the classification of the corresponding edge. In the figures that follow, we use "+" and "−" to label convex and concave edges, respectively. An occluding edge is labeled by "→"; when one moves in the direction of the arrow, the faces that meet at the edge are on the right side of the edge. Fig. 1 is an example of a labeled line drawing.

In the line drawing of an object, we refer to every point where the projections of edges meet as a junction. A junction is either the projection of a vertex of the object—a *vertex-junction*—or a point where the projections of nonadjacent edges meet—a *T-junction*.

From the labeled line drawing we construct the *labeled image structure graph* (LISG). It is an undirected graph augmented by labels on its nodes and arcs. For each im-

Manuscript received January 12, 1988; revised September 14, 1989. Recommended for acceptance by G. T Toussaint. This work was supported by the Semiconductor Research Corporation under Grant 88-11-008.

The authors are with the Computer Science Division, Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94720.

IEEE Log Number 8932050.

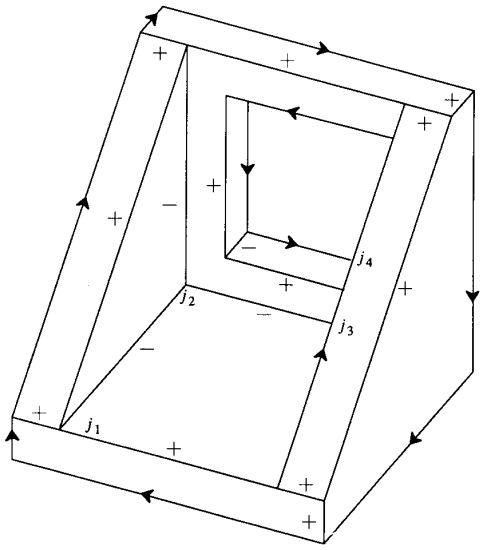


Fig. 1. A labeled line drawing. j_1 and j_2 are vertex-junctions; j_3 and j_4 are T-junctions.

age junction there is a node in the graph, and for each line segment in the image there is an arc between the nodes that corresponds to its endpoints. The arcs and the nodes are labeled by the labels of the corresponding line segments and junctions in the line drawing.

Definition: A viewpoint is *general* if there exists an open neighborhood of the viewpoint such that the LISG's that correspond to the line drawings of the scene, as viewed from points in this neighborhood, are all isomorphic to each other.

Intuitively, this means that from all points in the neighborhood of a general viewpoint the scene looks very similar: the lengths of lines and the angles between them may change but the basic graph structure remains the same.

Definition: A viewpoint that is not general is *accidental*.

B. The Viewing Data of an Object

The viewing space for orthographic projection is the space of viewing directions, which can be represented by a unit sphere where a point on the surface of the sphere corresponds to the direction vector with the same coordinates. Assume that an infinitely small, scaled-down, version of the object is placed at the center of the sphere. Orthographic projection with viewing direction p corresponds to viewing the object from point p on the sphere. We refer to this sphere as the *viewing sphere*, and to points on this sphere as the viewpoints of the orthographic projection.

The viewing sphere is partitioned into connected sets of points such that all points in a set have isomorphic LISG's, but the LISG's for points in adjacent sets are not isomorphic. We use the term *view* to refer to the representative LISG for a given set. Under this partition, the general viewpoints are grouped into open regions bounded by curves of accidental viewpoints. All viewpoints on a curve segment between adjacent regions have the same view. Where several regions share a boundary point, two or more boundary curves meet at a vertex. The view at the vertex is different from that of any of the viewpoints in its neighborhood. In other words, this partition has the structure of a planar graph embedded on the sphere, where the vertices, arcs, and faces of the graph are the vertices, curve segments, and regions of the partition, respectively.

As a viewer moves between a region and its boundary, or between a boundary curve and one of its endpoint vertices, the view changes—a *visual event* occurs.

Assuming that the viewpoint moves from region to region across boundary curves but it does not move along boundaries or across vertices, the aspect graph is the dual of the graph defined by the partition.

Following Callahan and Weiss [6], we define the *viewing data of an object* as the partition of the viewing sphere together with the view in each region of the partition. Fig. 2 shows the viewing data of an L-shaped object.

We can now precisely define the problem that is solved in this paper: compute the viewing data of an arbitrary polyhedral object.

III. Related Work

In the multiple 2-D view approach to 3-D object representation, the viewing space is partitioned into a finite number of regions, and a representative view is chosen for each region. In the matching stage of the object recognition process these views are matched against parts of the image. There are two approaches to partitioning the viewing space: 1) uniform, object-independent partitioning, and 2) partitioning of the viewing space into maximal regions of general viewpoints—the aspect graph approach.

A. The Uniform Partitioning Approach

In this approach the viewing space is partitioned in a uniform manner by projecting a tessellated regular polyhedron onto the sphere (see Fig. 3). The tessellation of the polyhedron and the positioning of the regions on the surface of the sphere is predetermined and independent of the structure of the objects. Examples of this approach are found in [10], [7].

The advantage of this approach is that the partition is easy to compute and fixed for all objects. The difficulty is in selecting the right scale of the partition. Under a fine partition, the views in many neighboring regions are qualitatively similar and provide no additional information for matching the views to the input image. On the other hand, under a coarse partition some important views might be missed. We illustrate this problem by examining the viewing data of the L-shaped object in Fig. 2. Under a fine uniform partition the large regions of the viewing data would be partitioned into many regions, while a coarse partition may miss the views in the small regions near the poles of the sphere.

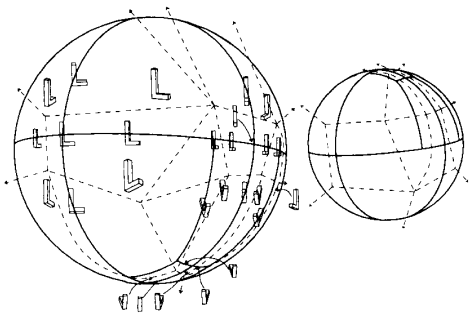


Fig. 2. The viewing data and the aspect graph of an L-shaped object. The partition is indicated by solid lines; the aspect graph by dashed lines. Representative views are shown for the regions, curves, and vertices of the partition in one hemisphere.

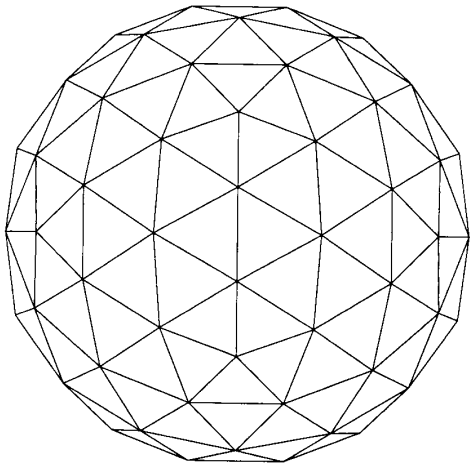


Fig. 3. The projection of a tessellated dodecahedron onto the viewing sphere.

B. The Aspect Graph Approach

In this approach, introduced by Koenderink and van Doorn [14], the viewing space is partitioned according to the qualitative structure of the view—the *aspect*. An aspect may be defined in one of several possible ways, depending on the application at hand. We have chosen one particular definition (Section II), namely the graph structure of the line drawing corresponding to an image of the object. Other definitions are possible: the aspect may be defined by the set of visible faces, by the set of occluding edges, or by any other topological measure on the view. The partition is defined by an equivalence relation on the set of viewpoints, where two viewpoints are equivalent if 1) the aspect from both viewpoints is the same, and 2) the two viewpoints are connected by a path of viewpoints along which the aspect does not change. The relation defines a partition of the viewing space into regions that correspond to equivalence classes of viewpoints. The structure of the partition is directly related to the structure and the visual complexity of the object. However, computing the partition is not a straightforward process.

For smooth objects the relation between the geometry of the object and location of the accidental viewpoints (the boundaries of the partition) is well understood. Some of the visual events and the location of the corresponding accidental viewpoints were first described by Koenderink and van Doorn [14]. A complete catalog of the visual events and the location of the corresponding viewpoints have been obtained [1], [2], [13]. Recently, Rieger [19] has published a catalog of the visual events for piecewise smooth objects that contain no planar edges. Callahan and Weiss [6] suggested the viewing data representation and gave examples of the viewing data of a few simple smooth objects. They did not provide a general algorithm for computing the viewing data of an object.

Chakravarty and Freeman [5] used this approach in the *characteristic views* representation of objects. In this representation, heuristic constraints on the orientation of the objects with respect to the camera are used for selecting a subset of the aspects as the representation of the object. The aspects and the partition of the viewing space are computed manually.

Hebert and Kanade [11] use the aspect graph approach for recognition of polyhedral objects in range images. They define an aspect by the set of occluding edges in the image. Ikeuchi [12] uses the aspect graph approach for representating objects in a system that uses photometric stereo. In this case, the aspects are defined by the faces that are detectable by photometric stereo. In both cases the exact partition of the viewing space is approximated by computing the set of views of a uniform partition, and then merging neighboring regions that have the same aspect. The uniform partition has to be fine enough to avoid missing any of the aspects (Hebert and Kanade [11] use about 2000 regions). As a result, many of the views belong to the same aspect and unnecessary computational cost is incurred in computing them. An additional cost is incurred in the comparison of views in adjacent regions of the uniform partition. When the aspect is defined by the set of visible features of the object, this is a relatively cheap operation. But for applications where the aspect is defined by the topology of the image structure graph, this operation is more expensive.

Werman, Baugher, and Gualtieri [22] present an algorithm for constructing the aspect graph of a convex polygon as viewed from viewpoints in the plane of the polygon, using perspective projection. Stewman and Bowyer [21] describe an algorithm for constructing the viewing data of convex polyhedra under perspective projection.

Plantinga and Dyer [16] presented the first algorithm for computing the exact viewing data of arbitrary polyhedral objects under orthographic projection. There were two major problems with that algorithm. The authors failed to recognize the visual event that results from the interaction of three nonadjacent edges, and it is therefore stated that the bound on the size of the partition of the viewpoint space is $O(n^4)$, and that the time complexity of the algorithm is $O(n^5)$, while the correct bounds are

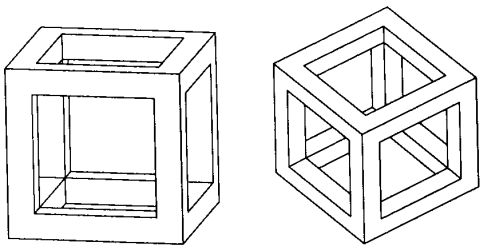


Fig. 4. Two line drawings that are qualitatively different, but the same faces are visible in both views.

$O(n^6)$ and $O(n^6 \log n)$. (This error was corrected in [17].) The second problem was that Plantinga and Dyer defined the aspect to be the set of visible faces of the object. Now, quite different line drawings may correspond to views where the same faces are visible (see Fig. 4). It is our opinion that this definition of the aspect is not appropriate for object recognition from line drawings extracted from intensity images.

Our work, first published in [9], was aimed at addressing these two problems. It is the first published algorithm for computing the viewing data for arbitrary polyhedra, where an aspect is defined by the graph structure of the line drawing. Contemporary to our work, Plantinga and Dyer [17], [18] have developed a modified and complete version of their algorithm, and an extension of the algorithm for computing the viewing data of polyhedral objects under perspective projection. In this work the definition of the aspect is closer to definition that is used in this paper.

Further comparison between this algorithm and the one due to Plantinga and Dyer may be found in Section VIII.

IV. The Locus of Accidental Viewpoints

In this section we describe how the structure of a polyhedral object is reflected in the location of the accidental viewpoints.

We will use the term *transparent* to refer to an object whose faces are transparent but its edges are not—it is the wire frame of the corresponding opaque object. The accidental viewpoints of an opaque object are a subset of those of the corresponding transparent object, as those accidental viewpoints that correspond to visual events of hidden parts of the objects are no longer relevant. Therefore, the partition for the transparent object (henceforth the T-partition) is a refinement of the partition for the same opaque object (henceforth the O-partition). A boundary segment of the T-partition is part of the O-partition if and only if the features that participate in the corresponding visual event are visible in one of the regions that are adjacent to that boundary. In this section we describe the accidental viewpoints in the transparent case. In Section VI we describe how to decide which of the accidental viewpoints of the transparent case are relevant to the opaque case.

In a visual event the structure of the LISG changes. The changes are either distinct junctions merging to a single junction, or junctions moving onto line segments. Recall that the LISG has three types of features: 1) vertex junctions, 2) T-junctions, and 3) line segments that are the projections of parts of edges. We therefore get the following events:

1) Two vertices project onto the same point.

2) A vertex and a T-junction project onto the same point.

3) A vertex projects onto the projection of a nonadjacent edge.

4) A T-junction projects onto the projection of another edge. The projections of three edges intersect at a point and therefore this is also the case where three T-junctions share the same point in the image. We assume that the three edges are skew to each another; the cases where either two or all three of the edges are coplanar reduce to one of the previous cases.

5) Parts of two line segments overlap. In this case at least one of the endpoints of one segment projects onto the other segment, and therefore this case is subsumed by cases 1 and 3.

Several of these events may combine to a single event, but that corresponds to the intersection of the loci of viewpoints corresponding to the component events. As every vertex has at least two non-collinear edges adjacent to it, cases 1 and 2 can be considered as limiting cases of 3 and 4, respectively. We can therefore regard cases 3 and 4 as fundamental, and need only study the locus of viewpoints from which a given vertex projects onto the projection of a given edge, and the locus of viewpoints from which the projections of three given edges intersect at a point.

A. The Interaction of a Vertex and an Edge

Let the vertex and the edge be v and $e = (a, b)$, respectively. v projects onto the projection of e only when the viewing direction is a convex linear combination of the vectors $a - v$ and $b - v$, or a convex linear combination of $v - a$ and $v - b$. On the viewing sphere, these directions are on two diametrically opposite arcs of a great circle. We define the arc from $a - v$ to $b - v$ to be the *front* arc and the other arc to be the *back* arc. See Fig. 5. Illustrations of the changes in the view that may take place upon crossing such a front or back arc may be found in Figs. 8 and 9.

When the edge and the vertex are shared by a face, the corresponding arcs are part of the great circle of viewing directions that are parallel to the plane of the face. This circle divides the sphere into two hemispheres: the *northern* hemisphere of viewpoints where the face is visible, and the *southern* hemisphere where it is invisible. The collection of arcs that correspond to all pairs of edges and vertices of the same face spans the whole circle. Therefore, each face induces a complete great circle—a *boundary circle*. The view changes associated with crossing a boundary circle are illustrated in Fig. 10.

B. The Interaction of a T-junction and an Edge

A T-junction is the location where the projections of two nonadjacent edges intersect. Therefore, the case

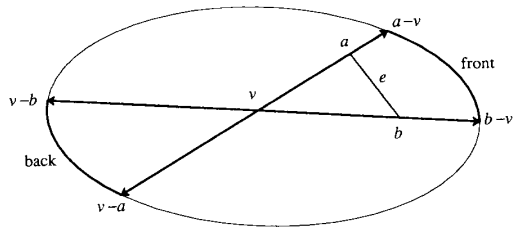


Fig. 5. The arcs of accidental viewpoints for the interaction between a vertex and an edge.

where a T-junction projects onto an edge is also the case where the projection of three nonadjacent edges intersect at a point. Assume, for a moment, that we are using a perspective projection. Under this assumption, the projections of three edges intersect at a point when a line of sight goes through all three edges. The viewpoint must therefore lie on the ruled surface that is defined by the family of lines that go through the three given edges. In the Appendix we show that this is a ruled quadric surface. As the edges are of finite extent, only parts of this surface actually contain viewpoints that are accidental with respect to these three edges.

Under orthographic projection, the accidental viewpoints lie on two antipodal curves that are defined by the direction vectors of the family of lines that pass through the three given edges. In the Appendix we show that these curves are either two "bent" ellipses or two great circles, and that they can be computed analytically. As in the perspective case, since the edges are of finite extent only parts of these curves actually contain viewpoints that are accidental with respect to these edges.

A line of sight that intersects all three edges defines a depth ordering of the edges with respect to the accidental viewpoint. As the edges do not intersect, this depth ordering is the same from all viewpoints on a continuous part of the boundary.

In the rest of the paper the term front/back *EV-boundary* refers to the locus of viewpoints from which a vertex projects on top of an edge, the term *boundary circle* refers to the locus of viewpoints where the visibility of a face changes, and the term *EEE-boundary* refers to the locus of viewpoints from which the projections of three edges intersect at a point. A *boundary segment* is a general term that refers to a part of a boundary curve between two adjacent vertices of the partition.

V. Overview of the Algorithm

To simplify the presentation, we assume first that a single visual event occurs at each boundary segment. At the end of this section we describe how to modify the algorithm to remove this restriction.

The algorithm computes the view of the opaque object (henceforth the O-view) in each region of the partition. It turns out that, for reasons of efficiency, the algorithm has to maintain the *augmented view of the transparent object* (henceforth the T-view). Roughly speaking, this is a view of the transparent object which is augmented by additional visible surface information. A detailed description of this data structure is presented in Section V-A. The outline of the algorithm is as follows.

1) Compute the T-partition:

- Compute boundary circles for all faces of the object, EV-boundaries for all pairs of edges and vertices that are not part of the same face, and EEE boundaries for all triplets of edges that are skew to each other.
- Compute the intersection of each boundary with all other boundaries and sort these intersections along the boundaries, merging intersections that correspond to the same point. As the intersections are computed, construct the graph structure of the T-partition incrementally, and merge boundary segments that coincide.

2) Pick an arbitrary region of the partition and compute the O-view and the T-view of the object, as seen from viewpoints within this region.

3) Traverse the partition in order of adjacent regions. At each boundary segment:

- Use the visible surface information in the T-view of the current region to test whether the boundary is part of the O-partition. If so, update the O-view according to the visual event that occurs at the boundary and store the new view. Else, remove the boundary and merge the two regions.
- Update the T-view according to the visual event that occurs at the boundary.

Continue this process until all boundaries have been examined.

So far, we have assumed that each boundary segment corresponds to a single event. To account for multiple events, each boundary segment points to a list of events associated with that boundary. We consider each event in the list separately, update the T-view, and remove events that are found to be invisible. If the list becomes empty the segment is removed. Else, we update the O-view according to the events that are left in the list. We can examine each event at a boundary segment independently, because accidental viewpoints where more than three features of the LISG interact simultaneously—several visual events combine to a single event—are confined to isolated viewing directions (e.g., the direction from which two or more vertices project onto single point). Such points do not occur inside a boundary segment; they are associated only with the vertices of the partition.

In the next subsection we describe the augmented view of the transparent object. In Section VI we provide a more detailed description of step 3 of the algorithm.

A. The Augmented View of the Transparent Object

We need to efficiently determine whether events are visible. It turns out that keeping track of faces whose projections are adjacent to each arc in the view—they are *visually adjacent* to that arc—is adequate for this purpose. To see this consider the following.

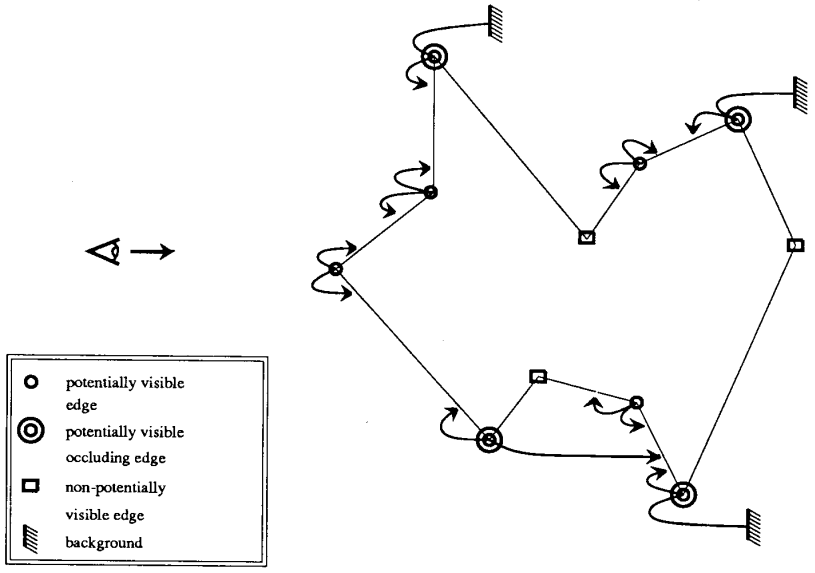


Fig. 6. The *augmented transparent view*. Shown is a planar slice through the object together with pointers to the faces that are visually adjacent to each edge.

A visual event is visible either when 1) all the features that participate in the event (an edge segment and a vertex, or three edge segments) are visible in the current region, or when 2) one feature $\mathcal{F}$ is currently hidden but will be visible in the next region. In the latter case, to decide whether an event is visible we have to decide whether $\mathcal{F}$ can become visible in the next region. In the next region $\mathcal{F}$ is not occluded by the occluding faces that are adjacent to visible features of the events (henceforth the *vfe*'s). However, $\mathcal{F}$ might be occluded by a face f_0 that is between it and the vfe's. Arbitrarily close to the boundary, the projections of $\mathcal{F}$ and the vfe's are arbitrarily close to each other, and therefore for f_0 to occlude $\mathcal{F}$ its projection must be adjacent to the projections of the vfe's. Since the event does not involve edges or vertices of f_0, its projection and the projections of the vfe's must also be adjacent in the current region.

The information about the faces that are visually adjacent to each visible arc is therefore sufficient to decide whether an event is visible. It turns out, however, that to efficiently update the visual adjacency information, whenever the view changes, it is necessary to keep additional information. Consider an event where an invisible occluding edge segment becomes visible (Fig. 7(b) to (a), for example). In some cases the only way to determine the face that is occluded by this segment (and is therefore visually adjacent to it) is by an exhaustive search of all the faces of the object. To avoid this search we maintain the *augmented view of the transparent object* (T-view) which is described below.

Define an edge to be *potentially visible* if it is not occluded by the interior of the object in the immediate neighborhood of the edge—along the line of sight, points that are arbitrarily close to the edge and are in front of it with respect to the viewpoint are outside the object. The T-view is a view of the transparent object in which each potentially visible arc points to the faces that are visually adjacent to this arc when all the parts of the object that occlude the corresponding edge are removed. See Fig. 6.

VI. The Visual Events of an Opaque Object

In this section we describe how to decide whether a boundary segment of the T-partition is part of the O-partition, given the O-view and the T-view in the current region, and the visual event that is supposed to occur at that segment. We also describe how to update the O-view to reflect the visual event. The update of the T-view is very similar and is therefore omitted.

A. Visual Events at an EEE-Boundary

At each segment of an EEE-boundary the three corresponding edges are classified as front, middle, and back according to their depth ordering from viewpoints on the boundary segment. A visual event occurs at the segment if and only if in the current view:

- The front and middle edges are occluding edges, and their projections meet in a T-junction.
- From viewpoints on the current boundary segment, the back edge is not behind the face that is occluded by the front and the middle edges at their T-junction.

When these conditions are satisfied, one of the following cases applies:

The back edge has a T-junction with each of the other edges [Fig. 7(a) to (b)]. The back edge becomes invisible. The segment between the T-junctions, and the T-junctions themselves are deleted.

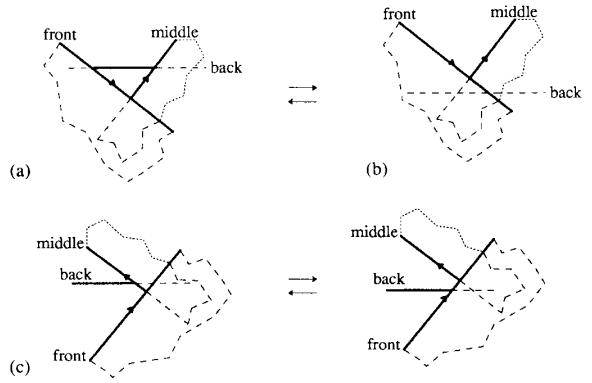


Fig. 7. The visual events at an EEE-boundary.

The back edge is invisible [Fig. 7(b) to (a)]. This is the inverse of the previous case.

Only one of the other edges has a T-junction with the back edge [Fig. 7(c)]. The T-junction moves to the other edge.

B. Visual Events at EV-Boundaries

An EV-boundary, which corresponds to a visual event where a vertex v and an edge e interact, is part of a great circle that divides the viewing sphere into two hemispheres. Upon crossing the boundary the viewpoint moves from the *current* hemisphere to the *next* hemisphere. We classify the edges that are adjacent to v according to the vector pointing from v to the other vertex of the edge:

C-edges: Edges whose vector is in the current hemisphere.

N-edges: Edges whose vector is in the next hemisphere.

B-edges: Edges whose vector is on the great circle.

1) Visual Events at a Front EV-Boundary: At a segment of a front EV-boundary,[1] if e is not an occluding edge, v is and remains invisible and this boundary segment is not part of the O-partition. Otherwise, an event that changes v's visibility may occur. There are two possible cases (see Fig. 8):

***V* becomes invisible.** The changes in the visibility of the edges adjacent to v are [Fig. 8(a) to (b) and (c) to (d)]:

- Visible N-edges become invisible.
- C-edge segments adjacent to v create T-junctions with e.
- B-edges become partially or fully occluded by the occluding face at e. Let $\mathfrak{Z}$ be the set of segments that form T-junctions with e. The segments of the B-edge whose visibility changes are those for which at least one end is adjacent to a segment in $\mathfrak{Z}$. If only one end is adjacent to a segment in $\mathfrak{Z}$, then the B-edge segment become partially visible and forms a T-junction with an edge that is adjacent to e in the occluding face at e [e.g., s_1 and s_2 in Fig. 8(a)]. If the other end is also adjacent to a segment in $\mathfrak{Z}$, then the B-edge segment becomes invisible (e.g., s_3 and s_4).

[1]Recall, that from a front (back) EV-boundary e is in front (back) of v with respect to the viewpoint (see Section IV-A).

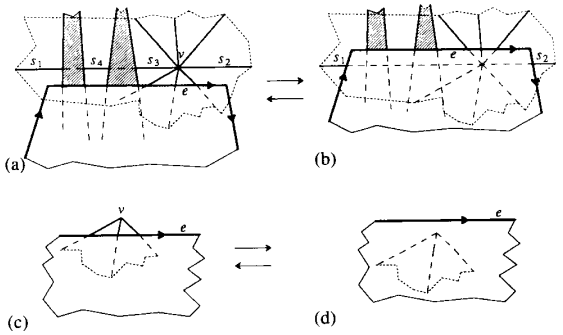


Fig. 8. Visual events at a front EV-boundary.

***V* becomes visible.** There are two possible cases:

- *Some N-edges have T-junctions with e* [Fig. 8(b) to (a)]. Partially visible N-edges become completely visible at v, B-edges become visible at v. Potentially visible C-edges become partially visible at e and create T-junctions with e. The junctions are inserted according to the ordering of the edges around v.

  B-edges have to be checked for partial occlusions by any face that is adjacent to the projection of e, and is not the occluding face at e.
- *No edge adjacent to v is visible at e* [Fig. 8(d) to (c)]. Let $\mathfrak{S}$ be the T-view segment of e that, in the visual event, coincides wth the projection of v. If $\mathfrak{S}$ is not visible then the event is not visible either. Else, the event is visible only if, from the boundary segment, v is in front of the occluded face at $\mathfrak{S}$. If the event is visible, then C-edges become partially visible and each creates a T-junction with $\mathfrak{S}$.

2) Visual Events at a Back EV-Boundary: At a segment of a back EV-boundary the event is visible only when v is visible, has occluding edges adjacent to it, and e is in front of the face that is occluded at v. Depending on the occluding edges at v, one of the following cases applies:

Both occluding edges are C-edges:

- The angle between the occluding edges, as measured along the projection of the faces that are adjacent to v, is less then π: e becomes fully visible at v (Fig. 9(a)→). The T-junctions of e with the occluding edges at v merge and disappear.
- The angle between the occluding edges is greater than π: e becomes occluded at v (Fig. 9(b)→). The visible segment of e between the occluding edges at v is removed.

Both occluding edges are N-edges: The events that are the inverse of the events described above occur: in-

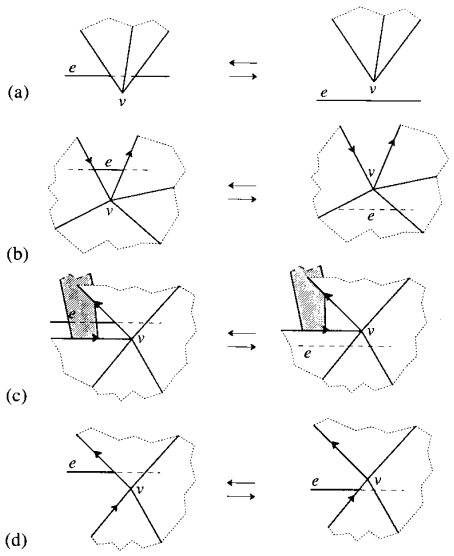


Fig. 9. The visual events at a back EV-boundary.

stead of e becoming visible, it becomes invisible and vice versa (Fig. 9(a) and (b)←).

One occluding B-edge: According to the other occluding edge, the visual events are almost the same as the ones described above. The difference is that, as in the event the projections of e and the B-edge coincide, the visibility of more than one segment of e may change. We identify these segments using a method similar to the one for checking the change in visibility of B-edge segments at a front EV-boundary (Section VI-B-1, the "V becomes invisible" case), and update the O-view accordingly [Fig. 9(c)].

One occluding C-edge and one occluding N-edge: The T-junction of e with the occluding edge at v moves to the other occluding edge at v [Fig. 9(d)].

C. Visual Events at a Boundary Circle

At a segment of a boundary circle that corresponds to a face f, a visual event occurs if and only if there is at least one edge of f that is partially visible in the current view. When the event occurs, f becomes either visible or invisible, depending on whether the viewpoint moves from the southern hemisphere to the northern hemisphere or vice versa. Below we describe the visual events that may occur when the viewpoint crosses the circle from north to south. When the viewpoint travels in the opposite direction, these events are reversed.

1) Crossing the Circle from North to South: The current view is in the northern hemisphere and parts of edges of f are visible. Fig. 10 illustrates the changes that occur in the view. We are considering the transition from the left view to the right view.

Edges of f:

1) Every visible part of a concave or an occluding edge of f becomes invisible. Examples are edges e_1 and e_2.

2) Every visible part of an inner contour of f becomes invisible. An example is e_{12}.

3) For visible segments of convex edges of the outer contour there are two possible cases:

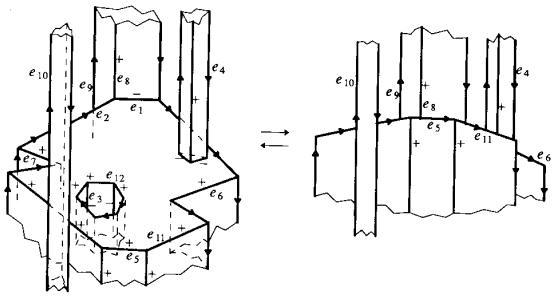


Fig. 10. The changes in the view when the visibility of a face changes.

a) The segment does not change its visibility, and becomes an occluding segment. An example is e_5.

b) The segment becomes partially or fully occluded by visible faces that are adjacent to f along convex edges of the outer contour. The part that remains visible becomes an occluding segment. This segment forms a T-junction with an edge of the occluding face (this is in addition to T-junctions it may form as an occluding segment). In the figure, e_6 becomes partially visible and occluding, while e_7 becomes invisible.

In a later part of this section we describe how to compute the change in the visibility of the convex segments of the outer contour.

Edges that are not part of f:

4) Visible edge segments that are adjacent to vertices of f:

a) An edge segment that is on the northern side of f forms a T-junction with an edge of f. Edges e_4 and e_8 are examples of this case. (In a later part of this section, we describe how to find the edge of f that forms the T-junction.)

b) Edge segments that are on the southern side of f and are adjacent to vertices of inner contours of f become invisible. Edge e_3 is an example of this case.

5) For edges that currently form T-junctions with edges of f there are two cases:

a) Edges that are occluded by an edge of f (e_9, for example) remain occluded, but the T-junction moves to a different edge of f.

b) Edges that are occluding f (e_{10}, for example) lose the T-junction with a segment of an edge of f that becomes invisible.

As discussed in case 3, for each visible convex edge segment of the outer contour we need to determine whether it remains fully visible, becomes partially visible or invisible. Observing that the depth ordering of the edges, as seen from viewpoints along the boundary segment, reflects the visibility of edges in the next view, we determine the visibility of convex edge segments of the outer contour of f by solving a 2-D hidden line removal problem (see Fig. 11). The lines are in the plane of f, and the image line is orthogonal to the viewing direction from an arbitrary point on the current boundary segment. This problem can be solved in a time that is linear in the number of edges, as the lines are part of a simple polygon (the outer contour of f).

In solving the above problem, we also order the visible

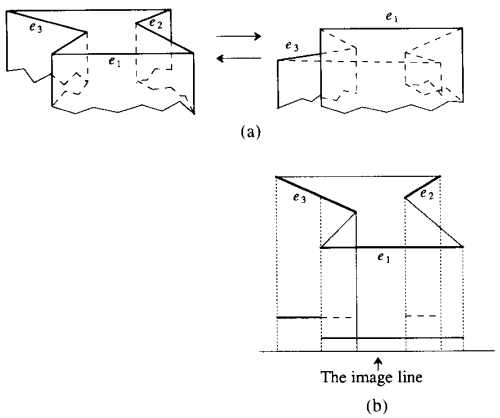


Fig. 11. The visibility of edges of the outer contour as a hidden line removal problem (a). The view of the face before and after crossing the boundary (b). The situation in the plane of the face.

segments along the image line. From the current O-view we form a sorted list of the edges that form new T-junctions with the edges of f (case 4a). Using the same viewpoint, the coordinates of the intersections of the projections of these edges with the image line are computed. As the list of the edges is already sorted, the T-junctions can be inserted into the segments of f in time that is linear in the number of segments and T-junctions.

To complete the update of the O-view in the next region, we remove all segments and junctions that become invisible.

VII. Complexity Analysis

Let the number of vertices of the object be n. The object has $O(n)$ edges and $O(n)$ faces. In constructing the T-partition, $O(n^2)$ EV-boundaries are computed in $O(n^2)$ time. To compute the EEE-boundaries every triplet of edges is considered in $O(n^3)$ time. In the worst case, $O(n^3)$ EEE-boundaries are produced. The $O(n)$ boundary circles are computed in $O(n)$ time.

To compute the T-partition, every boundary is intersected with all other boundaries. As there are $O(n^3)$ boundaries and computing the intersection of two boundaries takes constant time, the number of intersections and the time to compute them is bounded by $O(n^6)$; the intersections are sorted along the boundaries in $O(n^6 \log n)$ time. Therefore, the upper bounds on the time for computing the T-partition and the size of the T-partition are $O(n^6 \log n)$ and $O(n^6)$, respectively.

In an e-v event, T-junctions between e and edges adjacent to v are created or removed in $O(n)$ time. As the number of EV-boundary segments is $O(n^5)$, the total time for view updates at EV-boundaries is bounded by $O(n^6)$. As the number of EEE-boundary segments is bounded by $O(n^6)$ and updating the view at an EEE-boundary takes constant time, the time spent in updates for EEE events is bounded by $O(n^6)$. A face can have at most $O(n^2)$ T-junctions associated with it. As the number of segments of boundary circles is bounded by $O(n^4)$, the time spent in updating for events at segments of boundary circles is bounded by $O(n^6)$.

Summing up, the upper bound on the time to compute the opaque partition is $O(n^6 \log n)$. As the upper bounds on the size of a single view and the number of views are $O(n^2)$ and $O(n^6)$, respectively, representing each view explicitly, the worst case upper bound on the size of the viewing data is $O(n^8)$. While the time to compute all the changes that occur between views is bounded by $O(n^6)$, we have to incur $O(n^2)$ time in outputting each new view, and therefore the total time of the algorithm is bounded by $O(n^8)$.

VIII. Conclusions

We have presented a complete catalog of the visual events that occur for polyhedral objects under orthographic projection. We have also described how the structure of the object is reflected in the locus of accidental viewpoints. Given the catalog of visual events and knowledge about the locus of accidental viewpoints, we have presented an algorithm for computing the viewing data of polyhedral objects.

Plantinga and Dyer [17] compute a four-dimensional "visibility volume" (5-D for the perspective case) for each face of the object, and then perform set theoretic operations on these volumes to produce the visibility volume of the object. The boundaries of the partition are computed by projecting the boundaries of this visibility volume onto the viewing sphere and finding their intersection points. Their algorithm computes the partition in $O(m \log m + n^5)$ time, where m is the number of vertices in the partition. The views are computed in $O(n^2 m)$ time and space.

Both algorithms have the same worst case time complexity, but our algorithm is less output sensitive in the average case. However, in our approach all geometrical computations are performed directly in $\boldsymbol{R}^3$ or $\boldsymbol{R}^2$ and we do not use data structures in higher dimensions. We believe that this approach provides better insight into the possible visual events and the interaction between the features that participate in the event and other parts of the object. The detailed analysis of the visual events was recently used in developing a new faster algorithm [8] for computing the aspect graph of polyhedral objects.

The worst case upper bounds on the number of views and the size of the viewing data (and of the aspect graph) are $O(n^6)$ and $O(n^8)$, respectively. These bounds are reached only for very special cases. We believe that for common objects in an industrial environment the actual size of the viewing data will be much smaller. Implementation of the algorithm will provide a better estimate of the complexity of the average case.

Appendix
The Locus of Viewpoints of the EEE Events

Let the three edges that participate in the event be e_1, e_2, and e_3. Let a_i and $\bar{d}_i$ be an endpoint and the direction vector of e_i, respectively. Assume that we are using perspective projection with viewpoint p. The normal to the plane $\mathcal{P}_i$ through e_i and p is $(p - a_i) \times \bar{d}_i$. $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$ have a common intersection line which is the line of sight that goes through e_1, e_2, and e_3 (see Fig. 12). As the

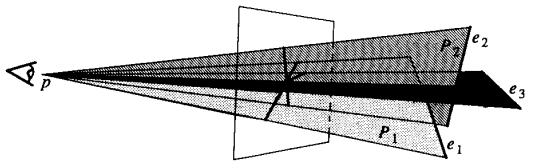


Fig. 12. The line of sight that goes through the three edges as the common intersection of three planes.

normals of all the planes are perpendicular to the intersection line, they are coplanar, and therefore:

$$[(p - a_1)\times\bar{d}_1\ (p - a_2)\times\bar{d}_2\ (p - a_3)\times\bar{d}_3] = 0. \tag{1}$$

This is a quadric equation in the coordinates of p, which defines a quadric ruled surface. It can be shown that when the edges are skew to each other, this is a hyperboloid of one sheet, and when all the edges are parallel to one plane, the surface is a hyperbolic paraboloid [20]. When two of the edges are coplanar the surface degenerates into a plane.

For an orthographic projection with viewing direction $[x, y, z]$, the viewpoint is of the form $\alpha[x, y, z]$, with α going to infinity. Substituting the viewpoint into (1) we get:

$$\lim_{\alpha\to\infty} \left(\alpha^2(c_1x^2 + c_2y^2 + c_3z^2 + c_4xy + c_5xz + c_6yz) + \alpha(c_7x + c_8y + c_9z) + c_{10}\right) = 0 \tag{2}$$

and therefore:

$$c_1x^2 + c_2y^2 + c_3z^2 + c_4xy + c_5xz + c_6yz = 0, \tag{3}$$

is the condition on the viewing direction of the EEE event in the orthographic projection. This is the equation of an elliptical cone, which in the case where (1) describes a hyperbolic paraboloid, degenerates into two intersecting planes. To get the curve of viewing directions on the viewing sphere we intersect the surface described by (3) with the sphere

$$x^2 + y^2 + z^2 = 1.$$

The resulting curves are either two antipodal "bent" ellipses, or, if (3) describes a degenerate cone, two great circles.

Acknowledgment

We thank J. Canny and H. Plantinga for useful discussions.

References

[1] V. I. Arnol'd, "Indices of singular points of 1-forms on a manifold with boundary, convolutions of invariants of reflection groups, and singular projections of smooth surfaces," *Russian Math. Surveys*, vol. 34, no. 2, pp. 1-42, 1979.

[2] ——, "Singularities of systems of rays," *Russian Math. Surveys*, vol. 38, no. 2, pp. 87-176, 1983.

[3] P. J. Besl and R. C. Jain, "Three-dimensional object recognition," *ACM Comput. Surveys*, vol. 17, no. 1, pp. 75-145, Mar. 1985.

[4] R. T. Chin and C. R. Dyer, "Model-based recognition in robot vision," *ACM Comput. Surveys*, vol. 18, no. 1, pp. 67-108, Mar. 1986.

[5] I. Chakravarty and H. Freeman, "Characteristic views as a basis for three-dimensional object recognition," in *Proc. Soc. Photo-Optical Instrumentation Engineers Conf. Robot Vision*, vol. 336, SPIE, Bellingham, WA, 1982.

[6] J. Callahan and R. Weiss, "A model for describing surface shape," in *Proc. IEEE Comput. Soc. Conf. Computer Vision and Pattern Recognition*, IEEE, 1985, pp. 240-245.

[7] G. Fekete and L. S. Davis, "Property spheres: A new representation for 3-d recognition," in *Proc. IEEE Workshop Computer Vision: Representation and Control*, IEEE, 1984, pp. 192-201.

[8] Z. Gigus, J. F. Canny, and R. Seidel, "Efficiently computing the aspect graph of polyhedral objects," in *Proc. Second Int. Conf. Computer Vision*, IEEE, Dec. 1988, pp. 30-39.

[9] Z. Gigus and J. Malik, "Computing the viewing data for polyhedral objects," in *Proc. 1988 IEEE Int. Conf. Robotics and Automation*, IEEE, Apr. 1988, pp. 1560-1566.

[10] C. Goad, "Special purpose automatic programming for 3d model-based vision," in *Proc. DARPA Image Understanding Workshop*, June 1983, pp. 94-104.

[11] M. Hebert and T. Kanade, "The 3d-profile method for object recognition," in *Proc. IEEE Comput. Soc. Conf. Computer Vision and Pattern Recognition*, IEEE, June 1985, pp. 458-463.

[12] K. Ikeuchi, "Precompiling a geometrical model into an interpretation tree for object recognition in bin-picking tasks," in *Proc. DARPA Image Understanding Workshop*, Feb. 1987, pp. 321-339.

[13] Y. L. Kergosien, "La famille des projections orthogonales d'une surface et ses singularités," *C. R. Acad. Sci. Paris*, vol. 292, pp. 929-932, 1981.

[14] J. J. Koenderink and A. J. van Doorn, "The singularities of visual mapping," *Biol. Cybern.*, vol. 24, pp. 51-59, 1976.

[15] ——, "The internal representation of solid shape with respect to vision," *Biol. Cybern.*, vol. 32, pp. 211-216, 1979.

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

[17] ——, "Visibility, occlusion and the aspect graph," Univ. Wisconsin—Madison, Tech. Rep. 736, Dec. 1987.

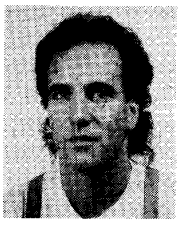
[18] W. H. Plantinga, "The Asp: A continuous, viewer-centered object representation for computer vision," Ph.D. dissertation, Univ. Wisconsin-Madison, Aug. 1988; also published as Comput. Sci. Tech. Rep. 784.

[19] J. H. Rieger, "On the classification of views of piecewise smooth objects," *Image and Vision Comput.*, vol. 5, no. 2, pp. 91-97, May 1987.

[20] G. Salmon, *Analytic Geometry of Three Dimensions*. Cambridge, England: Metcalfe, 1874.

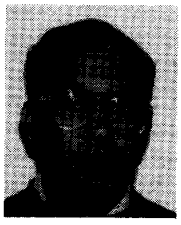
[21] J. Stewman and K. Bowyer, "Aspect graphs for convex planar-face objects," in *Proc. IEEE Comput. Soc. Workshop Computer Vision*, IEEE, Dec. 1987, pp. 123-130.

[22] M. Warman, S. Baugher, and J. A. Gualtieri, "The visual potential: One convex polygon," Center for Automation Research, Univ. Maryland, College Park, Tech. Rep. CAR-TR-212, Aug. 1986.



Ziv Gigus (S'85) was born in Petach Tikva, Israel, on April 3, 1957. He received the B.Sc. degree in mathematics and computer science from Tel Aviv University, Israel, in 1983 and the M.Sc. degree in computer science from the University of California at Berkeley in 1986.

He is currently pursuing the Ph.D. degree in computer science at the Department of Electrical Engineering and Computer Science, University of California at Berkeley. His research interests are in machine vision, computer graphics, and computational geometry.



Jitendra Malik (A'88) was born in Mathura, India, on October 11, 1960. He received the B.Tech degree from Indian Institute of Technology, Kanpur, in 1980 where he was awarded the gold medal for the best graduating student in electrical engineering. He received the Ph.D. degree in computer science from Stanford University, Stanford, CA.

Since January 1986, he has been an Assistant Professor in the Computer Science Division, Department of EECS, University of California at Berkeley. Since October 1988 he has also been a member of the group in Physiological Optics at UC Berkeley. His research interests are in machine vision and computational modeling of early human vision. These include work on edge detection, texture segmentation, line drawing interpretation, and 3-D object recognition.

Dr. Malik received a Presidential Young Investigator Award in 1989.