# Computing The Aspect Graph for Line Drawings of Polyhedral Objects [*]

Ziv Gigus and Jitendra Malik

## Abstract

Koenderink and van Doorn introduced aspect graphs as a way of representing 3-D shape for object recognition. 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. The viewing data of an object is the partition of the gaussian sphere together with a representative line drawings for each region of the partition. In this paper we present an algorithm to compute the viewing data of polyhedral objects. In the course of presenting the algorithm, we provide a full catalog of the visual events that occur for this type of objects.

[*]This research was supported by the Semiconductor Research Corporation grant number 82-11-008.

## Contents

## 1 Introduction

Three-dimensional object recognition is one of the main research areas in computer vision. Surveys of current approaches for solving the three dimensional object recognition problem may be found in Besl and Jain [BJ86] and Dyer and Chin [CD86]. One of the approaches that have been suggested is to compute a finite set of two dimensional views of the object from different viewpoints and match the image against this set.

Koenderink and van Doorn [KvD79] 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.

In this paper we present an algorithm for constructing the aspect graph for polyhedral objects under orthographic projection where an aspect is defined by the qualitative structure of the line drawing. In the course of presenting the algorithm we provide a full catalog of the visual events that occur for polyhedral objects.

## 2 The Labelled Image Structure Graph

We describe how to generate aspect graphs for polyhedral objects under orthographic projection. Objects can either be opaque or transparent; in the latter case, the objects are assumed to be made of tinted air—they do not refract light passing through them. We assume that lines in the line drawing of an object correspond only to depth and surface normal discontinuities.

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 meeting at the edge. In addition, a convex edge may be classified as an *occluding* edge: from the given viewpoint, both faces that meet along the edge are on the same side of the edge.

A *labelled line drawing* is a line drawing where each line has been labelled 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 labelled by "→"; when one moves in the direction of the arrow, both faces meeting at the edge are on the right side. Figure 1 is an example of a labelled line drawing.

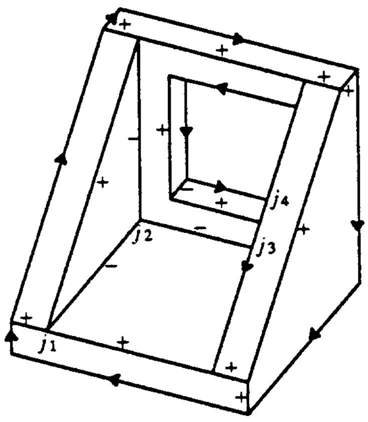


Figure 1: A Labelled Line Drawing. $j_1$ and $j_2$ are vertex-junctions, $j_3$ and $j_4$ are T-junctions.

In the line drawing of an object, we refer to every point where the projections of edges meet or intersect as a junction. The junctions are either the projection of vertices of the object—*vertex-junctions*, or points where the projections of non-adjacent edges meet. For transparent objects, we get junctions for vertices of the object and also for points where the projections of non-adjacent edges intersect. For opaque objects, we get a junction for every visible vertex—*vertex-junctions*, and for every point where the projection of a partially occluded edge meets the projection of the occluding edge. The latter junctions are classified as *T-junctions*.

From a labelled line drawing we construct the *labelled image structure graph* (LISG). It is an undirected graph augmented by labels on its nodes and arcs. For each image junction there is a node in the graph, and for each line segment in the image there is an arc between the nodes corresponding to its endpoints. The arcs and the nodes are labelled by the labels of the corresponding line segments and junctions in the line drawing.

**Definition.** The *extended LISG* is an LISG that includes explicit visible surface information by having each arc point to the faces of the object whose projections are adjacent to the corresponding line segment in the image.

**Definition.** A viewpoint is *general* if there exists an open neighborhood of the viewpoint such that the LISGs 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 interaction between features in the scene remains the same.

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

## 3 The Viewing Data of An Object

The viewing space of the orthographic projection is the space of viewing directions, and it can be represented by the gaussian sphere—a unit sphere where a point $p$ on the surface of the sphere corresponds to the direction vector with the same coordinates. We refer to this sphere as the *viewing sphere*. Assume that an infinitely small, scaled down, version of the object is placed at the origin of the gaussian sphere. Then the orthographic projection with viewing direction $p$ corresponds to viewing the object from point $p$ on the sphere. It is in this sense that we refer to a viewpoint of an orthographic projection.

The viewing sphere is partitioned into connected sets of points such that all the points in a set have isomorphic LISGs, but the LISGs 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 accidental viewpoints on a curve segment between adjacent regions have the same view. Where several regions share a boundary point, two or more curves meet resulting in 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, 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.

Note that if we assume that the viewpoint moves from region to region across boundary curves but does not move along the boundary or cross vertices, then the aspect graph is the dual of the graph defined by the partition.

Following Callahan and Weiss [CW85], we define the *viewing data of an object* as the partition of the viewing sphere together with the view at each region, curve segment and vertex of the partition. Figure 2 shows the viewing data of an L-shaped object.

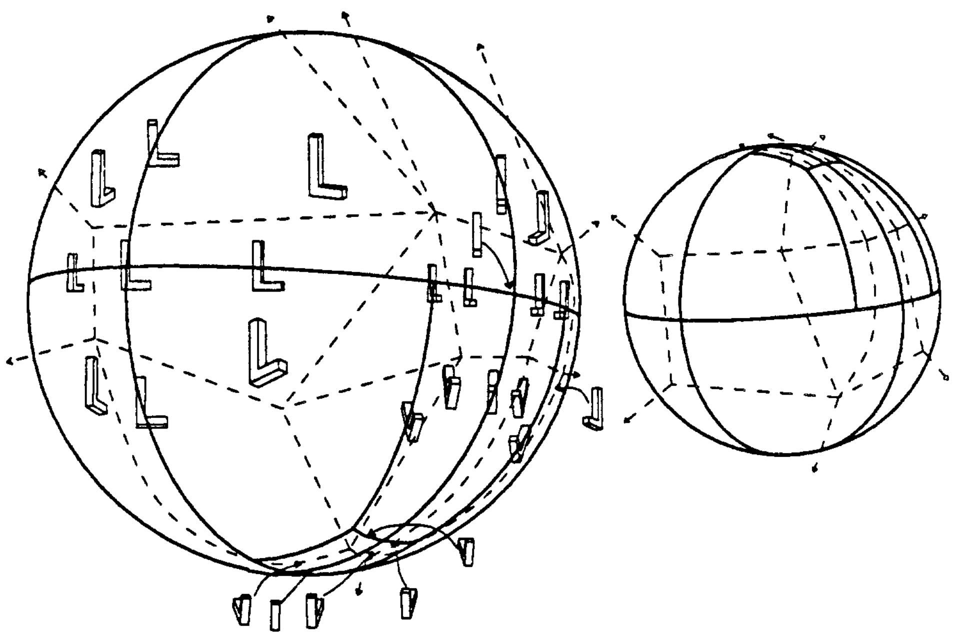


Figure 2: The viewing data and the aspect graph of an L-shaped object. The views are shown only in one hemisphere, the views on the other hemisphere are similar. In addition to the viewing data, the aspect graph is indicated by the broken lines.

## 4 Related Work

In an object representation scheme that uses the multiple view approach, the viewing space is partitioned into a finite number of regions, and a representative view is computed for each region. The parts of the image are then matched against this set of representative views. There are two approaches to the partitioning of 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.

### 4.1 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 Figure 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 Goad [Goa83], Fekete and Davis [FD84].

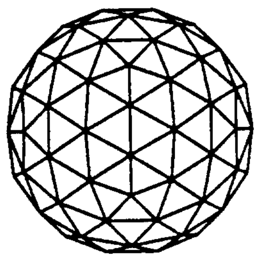


Figure 3: The projection of a tessellated dodecahedron onto the viewing sphere.

The advantage of this method is that the partition is easy to compute and it is fixed for all objects. However, the fact that the structure of the partition is independent of the geometric structure of the object is also the drawback of this approach. Under a fine partition, the views in many neighboring regions will be qualitatively similar and provide no additional information for the matching process. On the other hand, under a coarse partition some important views might be missed. We illustrate this problem by examing the viewing data of an L-shaped objects presented in Figure 2. Here, under a fine uniform partition the large areas of the viewing data would be partitioned into many regions with similar views, while a coarse partition may miss the views in the narrow regions.

## 4.2 The Aspect Graph Approach

The aspect graph approach, which has been introduced by Koenderink and van Doorn, is free from the drawbacks of the uniform partition approach. The main idea of this approach is to partition the viewing space according to the qualitative structure of the view–the *aspect*. The definition of the aspect is dependent on the particular application. For example, it may be defined by the topology of the image structure graph, by the set of visible faces, by the set of occluding edges, or by other non-metric properties of the view. The partition is defined by an equivalence relation on the set of viewpoints in which two viewpoints are equivalent if (1) the aspect from both viewpoints is the same, and (2) the two viewpoints are connect 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 advantage of this approach is that the partitioning is directly related to the structure and the visual complexity of the object. However, computing the partition is not a straightforward process. In fact, until recently there has been no published algorithms for computing the exact partition of the viewing space.

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 [KvD76]. A complete catalog of the visual events and the location of the corresponding viewpoints is provided in related papers [Arn79,Arn83,Ker81]. Callahan and Weiss [CW85] suggested the viewing data representation and used this catalog to give examples of the viewing data of a few simple smooth objects. However, they did not provide a general algorithm for carrying out this computation.

Chakravarty and Freeman [CF82] use the aspect graph 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.

Ikeuchi [Ike87] uses the aspect graph approach for representing objects in a system that uses photometric stereo. In this case, the aspects are defined by the faces that are detectable by photometric stereo. Kanade and Hebert [HK85] 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. 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 initial partition has to be fine enough so as to avoid missing any of the aspects (Hebert and Kanade use about 2000 regions). As a result, many of the computed views belong to the same aspect and therefore incur unnecessary computational cost. An additional cost is incurred by the comparison of views in adjacent regions of the uniform partition. For applications where an aspect is defined by the set of visible features of the object this is a relatively cheap operation. However, for applications where the aspect is defined by topology of the image structure graph this operation is more expansive.

Plantinga and Dyer [PD86] presented an algorithm for computing the aspect graph of polyhedral objects, where an aspect is defined by the visible faces of the object. This definition of an aspect may be appropriate for recognition of objects in range images, but it is not appropriate when line drawings are used. This is due to the fact that under this definition, line drawings that are topologically different may correspond to viewpoints that share a common aspect. For example, the two line drawings in Figure 4 are qualitatively very different, but the corresponding viewpoints share the same aspect.

## 5 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.

The LISG of an opaque object is a subgraph of the LISG of the corresponding transparent object, obtained by removing arcs and nodes that

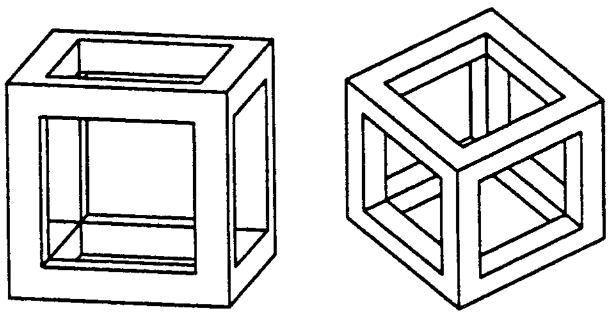


Figure 4: Two line drawings that are qualitatively different, but the same faces are visible in both views.

correspond to hidden parts of the object. At accidental viewpoints, special alignment of edges and vertices of the object results in a change in the structure of the LISG. It follows that the accidental viewpoints of an opaque object are a subset of those of the transparent object. Those accidental viewpoints that correspond to visual events of hidden parts of the object are not relevant in the opaque case. In this section we describe the accidental viewpoints in the transparent case. In section 8 we describe how to decide which of the accidental viewpoints of the transparent case are relevant to the opaque case.

As a viewer moves along a path of general viewpoints, the structure of the LISG does not change, but the distances between junctions and line segments in the image do change. A change in the structure of the LISG occurs when some of these distances go down to zero, resulting either in distinct junctions merging to a single point, or a junction moving onto a line segment. 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. The visual events correspond to interactions between these features, providing the following cases:

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 non-adjacent edge.

4. A T-junction projects onto of another edge. The projections of three

edges intersect at a point and therefore this is also the case in which three T-junctions share the same point in the image. (We assume that the three edges are skew to one 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 endpoint of the other segment, and therefore this case is subsumed by cases 1 and 3.

6. A combination of the above.

As every vertex has at least two non-collinear edges adjacent to it, cases 1 and 2 can be considered as limiting cases of cases 3 and 4, respectively. Case 6 is the intersection of the loci of viewpoints that corresponds to the other cases. Therefore, we need only study the locus of viewpoints where a given vertex projects onto the projection of a given edge, and the locus of viewpoints where the projection of three given edges intersect at a point.

## 5.1 The Interaction of a Vertex and an Edge

Let the vertex and the edge be $v$ and $e = (a, b)$ respectively, then $v$ 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 Figure 5.

Two observations will be of use later:

- 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. In the southern hemisphere, convex edges of the face are either occluding or invisible and concave edges are invisible. The collection of arcs that correspond to all edges and vertices of the same face spans the whole

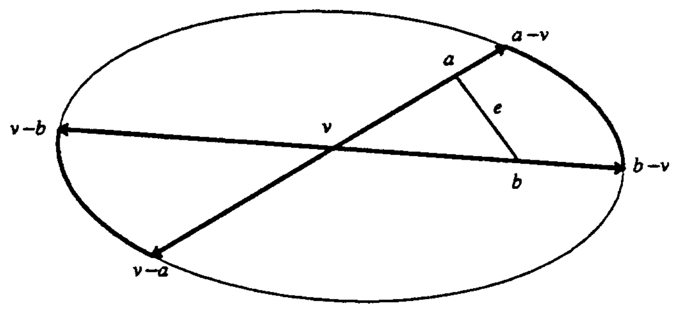


Figure 5: The arcs of accidental viewpoints for the interaction between a vertex and an edge.

circle. Therefore, for each face induces a complete great circle—a *boundary circle*.

- Given two edges, $(a_1, b_1), (a_2, b_2)$, the viewpoints from which the projections of these edges intersect are in two antipodal convex quadrilaterals whose boundaries are the arcs of accidental viewpoints where a vertex of one of the edges projects onto the projection of the other edge. This is illustrated in Figure 6.

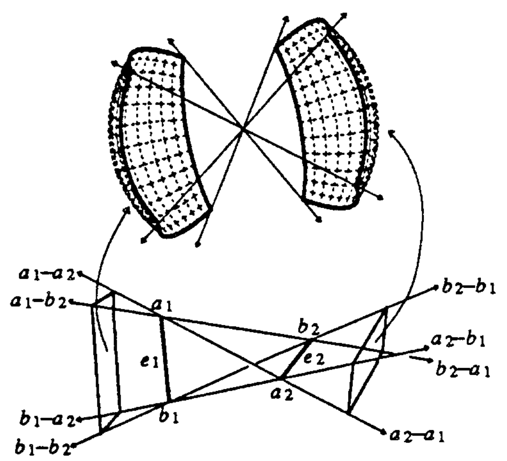


Figure 6: The quadrilaterals of viewpoints from which the projections of $e_1$ and $e_2$ intersect.

## 5.2 The Interaction of a T-junction and an Edge

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

In orthographic projection, the accidental viewpoints lie on a curve that is defined by the direction vectors of the family of lines that pass through the three given edges. In the appendix we show that, in the general case, this curve is quartic which can be computed analytically. Each pair of edges defines two quadrilaterals of viewpoints from which the projections of these edges intersect. Therefore, the regions where the projections of all three edges intersect are the intersection of three quadrilaterals corresponding to the three pairs of edges. Similar to the case of perspective projection, only the points that lie on the part of the curve that is contained in these regions are accidental points with respect to these three edges.

The line of sight that intersects all three edges defines a depth ordering of the edges with respect to the accidental viewpoint. In each of the quadrilaterals where the projections of a pair of edges intersect, the depth ordering of that pair with respect to the viewpoint along the line of sight through both edges is fixed. Therefore, in each region where the projections of all three edges intersect, the depth ordering of the edges at the accidental viewpoints is fixed.

In the rest of the paper the term front/back *EV-boundary* to refer to the locus of viewpoints from which a vertex projects on top of an edge, and the term *EEE-boundary* to refer to the locus of viewpoints from which the projections of three edges intersect at a point. *Boundary segment* is a general term that refers to a boundary between adjacent regions of general viewpoints—a boundary segment is any part of the boundary connecting two vertices of the partition.

## 6 Overview of the Algorithm

In the discussion that follows we assume that along each boundary segment only a single visual event occurs. This restriction does not apply in general and it is made only to simplify the presentation of the algorithm. At the end of this section we describe how to change the algorithm to remove this restriction.

For each boundary segment of the partition for the transparent object (henceforth T-partition) there is a corresponding edge and a vertex (for an *EV-boundary*) or three corresponding edges (for an *EEE-boundary*) that participate in the event occurring at the boundary. A boundary segment of the T-partition is part of the partition for the same opaque object (henceforth O-partition), if and only if the object features that participate in the event corresponding to that boundary are visible from one of the regions adjacent to this boundary. Suppose that we are given a boundary segment of the T-partition and the *extended view* of an opaque object on one side of this segment. We can use the visible surface information provided by the extended LISG, together with knowledge of the event that should occur at the segment, to decide whether this segment is part of the O-partition. If the segment is found to be part of the O-partition, we can compute the extended view in the adjacent region by updating the LISG according to the visual event that occurs along the boundary segment.

These observations lead to the following algorithm for computing the *viewing data* of an opaque object:

1. Compute the T-partition.

2. Pick an arbitrary region of the partition and compute the extended view of the opaque object, as seen from viewpoints within this region. For this step we can use any suitable hidden line removal algorithm.

3. Traverse the partition in order of adjacent regions. At each boundary segment, use the extended view in the current region to test whether the boundary is part of the O-partition. If the answer is positive, update the extended view according to the visual event that occurs at the boundary, otherwise remove the boundary and merge the two regions. Upon moving to the new region store the view that cor-

responds to that region. Continue this process until all regions are visited.

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, and remove events that are found to be invisible. When the list becomes empty the segment is removed, otherwise we update the extended 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 two features of the LISG interact simultaneously—several visual events combine into a single event—are confined to isolated viewing direction (e.g. the direction from which two or more vertices project onto single point), that is, these points are vertices in the partition.

In the next two sections we provide a more detailed description of the first and last steps of the algorithm.

## 7 Computing the T-Partition

The T-partition is computed as follows:

1. For each face of the object compute the corresponding boundary circle and make the circle point to the corresponding face.

2. Compute the EV-boundaries for all edges and vertices that are not part of the same face. With each EV-boundary store the triplet $< e,v,f\text{-}or\text{-}b >$, where $e$ and $v$ are pointers to the corresponding edge and vertex, and $f$-$or$-$b$ is a flag that indicates where it is a front or a back EV-boundary.

3. Compute the EEE boundaries. For each triplet of edges $(e_1,e_2,e_3)$ that are skew to each other, compute the regions of intersection of the rectangles that correspond to edge pairs, and then compute the curve segments that intersect each of these regions. With each EEE-boundary segment store the triplet $< e_i,e_j,e_k >$ of pointers to the corresponding edges ordered by depth, as seen from viewpoints along the segment.

4. Compute the intersection of each boundary segment with all other boundary segments and sort these intersections along each boundary, 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.

## 8 The Visual Events of An Opaque Object

This section describes how to decide whether a boundary segment of the T-partition is part of the O-partition, given the extended view in the current region and the information about the visual event that is supposed to occur at the segment. We also describe how to update the extended view to reflect the visual event that occurs when we move to the new region.

The following data structures are maintained:

- Static information about the object is kept in standard boundary representation data structure in which each face, edge and vertex is represented. Every face points to the list of contours used by that face, and every contour points to the edges used by that contour. Every edge points to its two end-vertices and to the two faces sharing this edge. In addition, every edge is marked as convex or concave. Every vertex points to a doubly-linked, circular list of edges that use it. The list is ordered such that, in the forward direction, adjacent edges are in clockwise order around the vertex. With each vertex we also store the coordinates of the vertex in the canonical reference frame.

- In addition to the static information described above, at every step of the traversal process, we maintain current visibility information of each feature of the object (this is in addition to the extended view). Every vertex is marked as visible or invisible. Every edge is marked as visible, partially visible or invisible. An edge that is partially visible points to the list of T-junctions on that edge. The order of the list corresponds to the order of the T-junctions along the edge going from one vertex to the other. We also use this list to keep visibility information about the segments of the edge between the T-junctions along the edge. A T-junction in the current extended view points back to its locations in the lists of the two edges whose projections form the T-junction.

This information is updated whenever the current view changes.

- We remind the reader that in the extended view, every segment points to the two faces whose projections are adjacent to the corresponding line segment in the image. If there is only one face of the object whose projection is adjacent to the segment (as is the case for some segments of occluding edge), the other face that the segment points to is the background "face", which is always at infinity with respect to the current viewpoint.

## 8.1 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.

### 8.1.1 Visual Events at a Front EV-boundary

At a segment of a front EV-boundary, 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. According to whether is visible or invisible in the current view there are two possible cases:

**$V$ becomes invisible.** The changes in the visibility of the edges adjacent to $v$ are (see Figure 7a):

- Any visible N-edge becomes invisible.

- C-edge segments adjacent to $v$ create T-junctions with $e$.

- B-edges become partially or fully occluded by the visible face adjacent to $e$. We check for changes in the visibility of all the visible segments of the B-edge, even those that are not adjacent to $v$ in the LISG. Segments whose visibility changes are:

  - Segments that become fully invisible. One end of such a segment is a T-junction with an edge that has a T-junction with $e$, and the other end is either $v$ (e.g. $s_1$) or a T-junction with another edge that has a T-junction with $e$ (e.g. $s_2$).

  - Segments that become partially occluded by the occluding face at $e$. Such a segment has only one end that is $v$ (e.g. $s_3$) or is a T-junction with another edge that has a T-junction with $e$ (e.g. $s_4$). The segment forms a T-junction with an edge that is adjacent to $e$ in the occluding face at $e$.

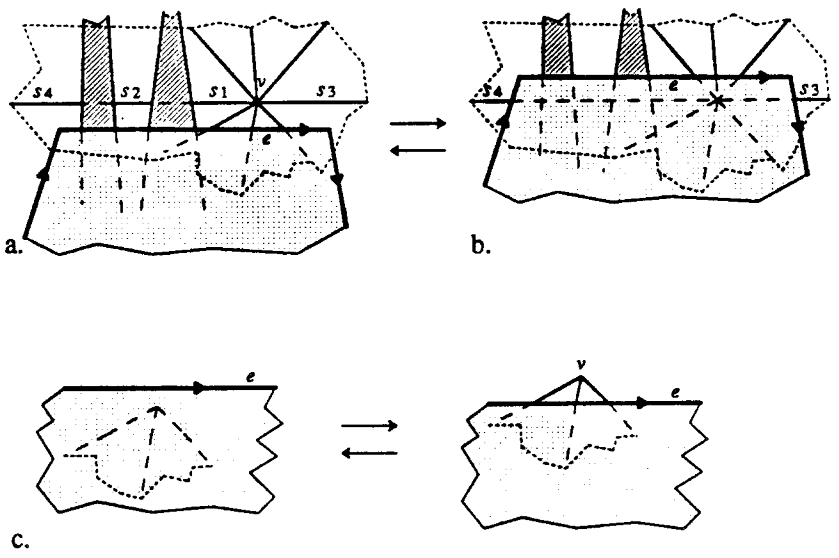


Figure 7: Visual events at a front EV-boundary.

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

- *Some N-edges have T-junctions with e* (Figure 7b). Partially visible N-edges become completely visible at $v$, B-edges become visible at $v$. C-edges that are adjacent to a face whose normal points in the direction of the current EV-boundary become partially visible at $e$ and create T-junctions with $e$.

  The junctions are inserted according to the ordering of the edges as seen from a point on the current boundary segment. If one of the N-edges is an occluding edge, then the first C-edge or B-edge that is counterclockwise from this edge and becomes visible is marked as an occluding edge.

  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* (Figure 7c). We have to determine whether $v$ becomes visible in the next view. In the next view, $v$ is not occluded by the occluding face at $e$, but might be occluded by faces that partially occlude $e$, or are occluded faces at $e$. The extend view determines whether $v$ is visible in the next view. We test The list of segments of the view that correspond to $e$, to find which segment coincides with the projection of $v$ when viewed from a point on the current EV-boundary segment. If there is no such segment then $v$ is not visible in the next view and the event is invisible. Otherwise, we compare $v$ with the occluded face at the given segment $e_i$ of $e$. The event is visible only when $v$ is in front of that face with respect to the viewpoint. If $v$ becomes visible, then the C-edges become partially visible and each creates a T-junction with $e_i$. See Figure 7c.

### 8.1.2 Visual Events at a Back EV-boundary

At a segment of a back EV-boundary $v$ projects on top of $e$ only when $e$ is in front of one of the faces whose projection is adjacent to the projection of $v$ in the current view. Otherwise, the event is invisible at this segment. When $e$ is visible we have one of the following cases:

**Both N-set and C-set are not empty.** There are several subcases:

- *There is an occluding B-edge* (Figure 8a). If the other occluding edge is a C-edge then $e$ becomes invisible at $v$. All the segments of the view that correspond to $e$ and are adjacent to segments of edges that form T-junctions with the B-edge has to be removed from the view.

  If the other occluding edge is an N-edge, then $e$ becomes visible at $v$. We check $e$ against all the faces whose projection is adjacent to the B-edge, but are not the occluding face at the edge, and add a segment to the view for each part of $e$ that is visible.

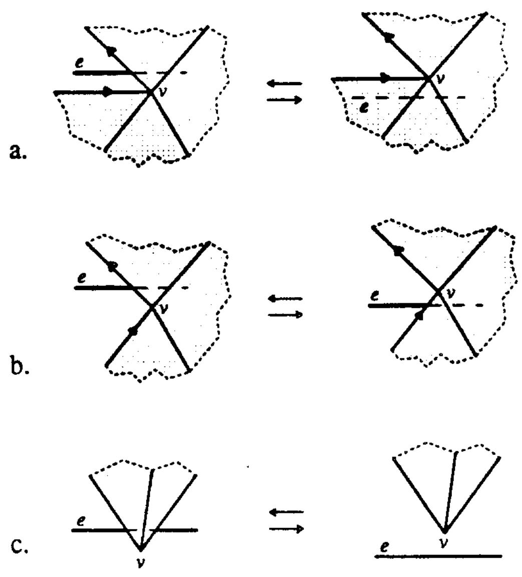


Figure 8: The visual events at a back EV-boundary.

- *There is no occluding B-edge.* The T-junction of $e$ with an occluding C-edge moves to the occluding N-edge. See Figure 8b.

**Either the C-set or the N-set is empty** .See Figure 8c.

- *The C-set is not empty–e* is partially occluded by the faces at $v$. T-junctions of $e$ with the edges at $v$ disappear. If there are B-edges then parts of $e$ that were occluded by the faces adjacent to these edges become visible.

- *The N-set is not empty.* $E$ gets T-junctions with the occluding N-set edges. If there are B-edges then the segments of $e$, both ends of which are T-junctions with edges that have T-junctions with the B-edges, are removed.

## 8.2 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 has visible parts 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.

### 8.2.1 Crossing the Circle from North to South

The current view is in the northern hemisphere and parts of edges of $f$ are visible. Figure 9 illustrates the changes that occur in the view:

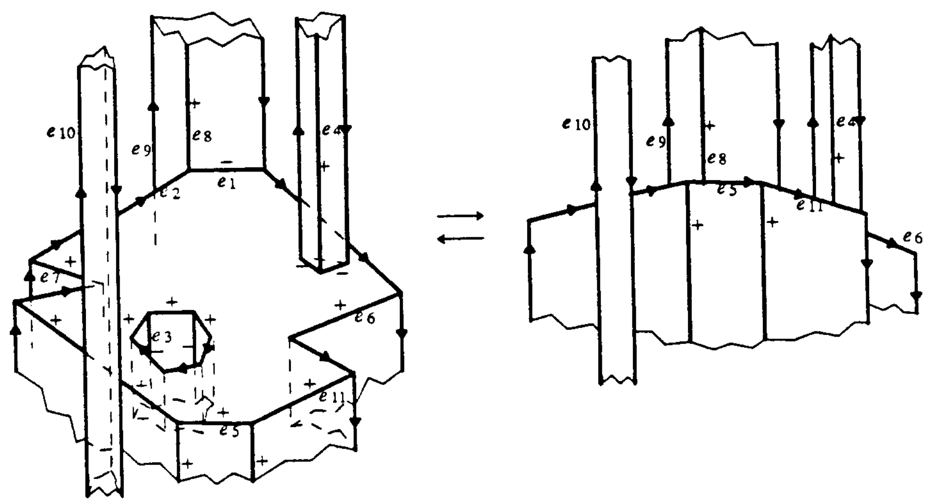


Figure 9: The changes in the view when the visibility of a face changes.

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.

3. For visible edge segments that are not part of $f$ but are adjacent to vertices of the inner contours there are two possible cases:

   (a) A segment that is on the southern side of $f$—part of a 'hole'—becomes invisible. Edge $e_3$ is an example of this case.

   (b) A segment that is on the northern side of $f$ ($e_4$, for example) forms a T-junction with an edge of $f$. (In a later part of this section, we describe how to decide where to insert this T-junction).

4. For visible segments of convex edges of the outer contour there are two possible cases:

   (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 of the segment 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.

5. Visible segments that are not part of $f$ and are adjacent to vertices of the outer contour of $f$ that become invisible form T-junctions with edges of $f$. Edge $e_8$ is an example for this case. (In a later part of this section, we describe how we decide where to insert the T-junction).

6. For edges that currently have T-junctions with edges of $f$ and are not adjacent to $f$ 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 from a current occluding edge of $f$ to an edge of $f$ that is currently convex. $E_9$ is an example of this case.

   (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.

The visibility of convex edge segments is determined as follows. Let $e_a$ be a visible convex edge that becomes fully or partially occluded, let $f_1$ be a face that is adjacent to $f$ and in the next view is a visible face that occludes $e_a$, and let $e_b$ be the edge that is shared by $f$ and $f_1$ (for example, in Figure 9 $e_6$ and $e_{11}$ correspond to $e_a$ and $e_b$ respectively). Then, from a viewpoint on the current segment of the boundary circle, $e_b$ is visible and it is in front of $e_a$, or equivalently $e_a$ is hidden (or partially hidden) behind $e_b$. It follows that the depth ordering of the edges, as seen from a viewpoint along the boundary segment, reflects the visibility of edges in the next view. As any change in the visibility of an edge is a visual event, the depth ordering of the edges of $f$ from all viewpoints along a given boundary segment must be the same.

Therefore, we can determine the visibility of segments of convex edges of the outer contour of $f$ by solving a problem of 2D hidden line removal. 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. As the lines are part of a simple polygon (the outer contour of $f$), this problem can be solved in a time that is linear in the number of edges. See Figure 10.

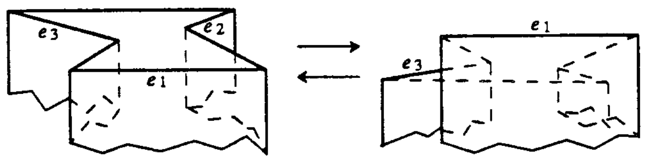
In solving the above problem, we also order the visible segments along the image line. From the current extended view we form an ordered list of the of edges that form new T-junctions with the edges of $f$ (cases 3b and 5). Using the same viewpoint the coordinates of the intersections of the projections of these edges with "image line" are computed. As the list of the edges is already ordered, the T-junctions can be inserted into the segments of $f$ in time that is linear time in the number of segments and T-junctions.

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

### 8.2.2 Crossing the Circle from South To North

The face becomes visible. The changes detailed in the previous section are reversed and the update of the extended view is done in a similar way.

a.



b.

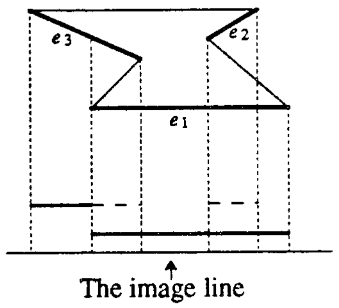


Figure 10: The visibility of edges of the outer contour of a face as a hidden line removal problem. a. The view of the face before and after crossing the boundary. b. The situation in the plane of the face.

## 8.3 Visual Events at an EEE-Boundary

At each segment of an EEE-boundary the three edges involved are classified as front, middle, or 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.
- Given the face that is the occluded face at the T-junction, the back edge is either shared by this face, or is in front of it with respect to viewpoints on the current segment.

If both these conditions are satisfied, there are the following cases:

**The back edge has a T-junction with each of the other edges** (Figure 11a to b). The back edge becomes invisible. The segment between the T-junctions, and the junctions are deleted.

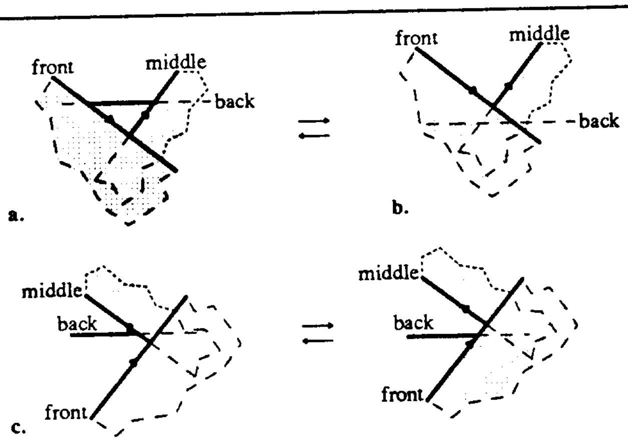


Figure 11: The visual events at a EEE-boundary.

**The back edge is invisible** (Figure 11b to a). This is the inverse of the previous case.

**Only one of the other edges has a T-junction with the back edge** (Figure 11c). The T-junction moves to the other edge.

## 9 Computing The Views at the Boundary of the Partition

The views along the boundary segments of the partition are computed as intermediate views between the current and the next view during the traversal of the partition, whenever a boundary is found to be part of the O-partition.

To compute the view at a vertex of the partition, we pick a view along an incident boundary and modify it according to the changes that correspond to moving onto each of the other boundaries that are incident upon the vertex.

## 10 Complexity Analysis

In this section we give a worst case upper bound on the complexity of the algorithm.

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

In step 4 of computing 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 time for sorting the intersections is bounded by $O(n^6 \log n)$. 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 edge-vertex event, T-junctions between the edge that corresponds to the boundary and edges adjacent to the vertex are created or deleted. Thus, $O(n)$ is the bound on the time for updating the extended view for this event. As the number of EV-boundaries is bounded by $O(n^2)$, their intersection with other boundaries may result in $O(n^5)$ EV-boundary segments. Therefore, the total time for updates of the view for events 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 total time spent in update for EEE events is bounded by $O(n^6)$. A partially visible face can have at most $O(n^2)$ T-junctions associated with it. As there are $O(n)$ boundary circles, the number of segments of boundary circles is bounded by $O(n^4)$. Therefore, the total time spent in updating for events at segments of boundary circles is bounded by $O(n^6)$.

Summing up, the upper bound on time complexity of the algorithm is $O(n^6 \log n)$. In the worst case, almost every edge creates a T-junction with every other edge, and therefore the worst case upper bound on the size of the LISG for a single view is $O(n^2)$. As the size of the partition is bounded by $O(n^6)$ the size of the complete viewing data of an object is bounded by $O(n^8)$.

## 11 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 object, which is an extension to the aspect graph.

A worst case complexity analysis indicates that the time complexity of the algorithm is bounded by $O(n^6 \log n)$, and that the size of this data structure (and of the aspect graph) is bounded by $O(n^8)$. We believe that for common objects in an industrial environment the actual size of the viewing data will be much smaller. We plan to implement the algorithm to get a better estimate of the size of the data structure and the complexity of algorithm in the average case.

The visual events for polyhedral objects and for objects bounded by single smooth surfaces has been fully cataloged. The algorithms for constructing the viewing data or the aspect graph for these classes of objects are also well understood. However, most objects in actual applications are actually piecewise smooth objects. Further research is needed to find algorithms for constructing aspect graphs for this class of objects.

## Acknowledgements

We thank John Canny for useful discussions. He pointed out that the accidental viewpoints for the triple edge case form a ruled quadric surface.

# Appendix

## The Locus of Viewpoints of the EEE Event

Let the three edges that participate in the event be $e_1$, $e_2$, and $e_3$. Let $a_i$ and $\vec{d_i}$ 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 $(p-a_i)\times\vec{d_i}$. $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$ has a common intersection line which is the line of sight that goes through $e_1$,$e_2$ and $e_3$ (see Figure 12). As the normals of all the planes are perpendicular to the intersection line, they are coplanar, and therefore:

$$[(p-a_1)\times\vec{d_1}\;\;(p-a_2)\times\vec{d_2}\;\;(p-a_3)\times\vec{d_3}] = 0. \tag{1}$$

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[Sal74]. When two of the edges are coplanar the surface degenerates into a plane.

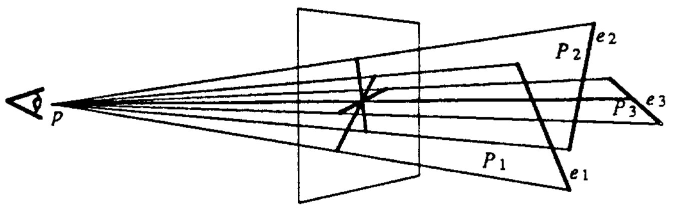


Figure 12: The line of sight that goes through the three edges as the common intersection of three planes.

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

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

and therefore:

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

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 Equation 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 Equation 2 with the sphere

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

The resulting curves are either two antipodal "bent" ellipses, or, in the case where Equation 2 describes a degenerate cone, the curves are two great circles.

## References

[Arn79] 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 Mathematical Surveys*, 34(2):1–42, 1979.

[Arn83] V. I. Arnol'd. Singularities of systems of rays. *Russian Mathematical Surveys*, 38(2):87–176, 1983.

[BJ86] P. J. Besl and R. C. Jain. Three-dimensional object recognition. *ACM Comput. Surv.*, 17(1):75–145, March 1986.

[CD86] R. T. Chin and C. R. Dyer. Model-based recognition in robot vision. *ACM Comput. Surv.*, 18(1):67–108, March 1986.

[CF82] I. Chakravarty and H. Freeman. Characteristic views as a basis for three-dimensional object recognition. In *Proceedings of The Society for Photo-Optical Instrumentation Engineers Conference on Robot Vision*, pages 37–45, Vol. 336, SPIE, Bellingham, Wash., 1982.

[CW85] J. Callahan and R. Weiss. A model for describing surface shape. In *Proceeding of the IEEE Computer Society Conference on Pattern Recognition and Image Processing*, pages 240–245, IEEE, N.Y., 1985.

[FD84] G. Fekete and L. S. Davis. Property spheres: a new representation for 3-d recognition. In *Proceedings of IEEE Workshop on Computer Vision: Representation and Control*, pages 192–201, IEEE, N.Y., 1984.

[Goa83] C. Goad. Special purpose automatic programming for 3d model-based vision. In *Proceedings of DARPA Image Understanding Workshop*, pages 94–104, June 1983.

[HK85] M. Hebert and T. Kanade. The 3d-profile method for object recognition. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 458–463, IEEE, N.Y., June 1985.

[Ike87] K. Ikeuchi. Precompiling a geometrical model into an interpretation tree for object recognition in bin-picking tasks. In *Proceedings of DARPA Image Understanding Workshop*, pages 321–339, February 1987.

[Ker81] Y. L. Kergosien. La famille des projections orthogonales d'une surface et ses singularitiés. *C. R. Acad Sc. Paris*, 292:929–932, 1981.

[KvD76] J. J. Koenderink and A. J. van Doorn. The singularities of visual mapping. *Biol. Cybern.*, 24:51–59, 1976.

[KvD79] J. J. Koenderink and A. J. van Doorn. The internal representation of solid shape with respect to vision. *Biol. Cybern.*, 32:211–216, 1979.

[PD86] W. H. Plantinga and C. R. Dyer. An algorithm for constructing the aspect graph. In *Proceedings of the 27th. Symp. on the Foundation of Computer Science*, pages 123–131, IEEE, N.Y., 1986.

[Sal74] G. Salmon. *Analytic Geometry of Three Dimensions*. W. Metcalfe printers, Cambridge, England, 1874.