Chapter 1

Introduction

Current approaches to the two broad areas of computer graphics and computer vision are related by common representations and imaging models. Our understanding both of the mathematics of 3D shape and of the computer modeling of shape has progressively and intimately joined graphics and vision together. The depth of this relationship and the implications for future work are only now beginning to be understood. This thesis presents a study of a new representational framework that addresses specific problems in computer vision and computer graphics. This framework for modeling the appearance of 3D shape relates the two problems of shape display for visualization and shape understanding for recognition.

A major theme through this work is the *visual event* [Koen76]. Very generally, an event is an image change resulting from a change in the viewer's vantage point. If the problem is to display a model from a set of viewpoints, as in animation, then the visual event is the change between frames in the animation sequence. For machine vision problems, the visual event is the change in the images that are gathered over time or from neighboring viewpoints. The relationship of the visual event to 3D shape and to viewpoint is one of the central problems studied in this thesis.

Shape representations have been broadly categorized previously as viewer-centered or object-centered. An object-centered representation encodes the volume of space that an object occupies independent of the projection mapping used to generate its appearance. A viewer-centered representation stores the appearance of an object from one or more viewpoints. Viewer-centered representations include information about appearance that is not explicit in object-centered models. The projection process creates strong visual cues that are not represented in object-centered models. The important advantage of the viewer-centered representation is that features of appearance such as occlusion can be made explicit to be used as part of the shape itself. This thesis presents a viewer-centered representation of shape and studies the benefits of using this representation for specific problems in computer graphics and computer vision.

In computer graphics, the problem of interactively viewing a static scene requires the computation and display of a sequence of images generated from a user-controlled, continuously-changing viewpoint. The goal is to display a 3D scene interactively as it would appear from a particular vantage point; the appearance of the scene must be updated dynamically as the user interactively modifies the vantage point within the environment. Our approach to this problem is to use the set of visual events computed in a viewer-centered representation to take advantage of the close similarity between adjacent frames in the display sequence. This coherence from frame to frame lends itself to a method of representing only changes between frames rather than computing a complete projection. The viewpoints where structural changes occur are computed through the construction of a representation that makes explicit exactly which vertices, edges and faces are visible as a function of viewpoint. The algorithm has two phases: a preprocessing phase, in which the initial appearance of the polyhedron and the visual events are computed, and an on-line, interactive phase, in which a sequence of frames along a user-controlled viewpath is displayed in real time.

This thesis shows that there are advantages over other methods in using visual events for solving the interactive viewing problem. First, exploiting the coherence between frames makes this a very efficient approach. The visible lines (or surfaces) of a polyhedral scene can be displayed for a sequence using this approach at a rate comparable to wire-frame display. Second, although there is a precomputation step, this is an exact, object-space method so that the display of the scene is exact no matter how small or large the changes in viewpoint may be. Third, unlike methods that totally precompute the set of images, there is flexibility in rendering. The visible line (surface) computation is separated from the rendering so that sequences can be viewed under different shading and lighting conditions (or with only visible lines) without the need to repeat the precomputation step of the algorithm.

In computer vision, the problem of how to solve for the set of viewpoints from which a 3D model will project to an observed set of shape features is considered an important step in certain object recognition paradigms [Grim90, Lowe87]. Solutions to the viewpoint-determination problem are based on how features of 3D objects will appear in an image. Our approach to this problem is to use as a model a precomputed representation of the appearance of the occluding contour. This viewer-centered representation is essentially the set of visual events that affect the occluding contour. These visual events, such as T-junctions, are strong cues in the projection of 3D shape

[Rich88]. Features of the occluding contour are stored and organized into a structure making inter-feature relationships and dynamic feature changes explicit. This approach can be distinguished from previous methods in two respects: (1) the kinds of features that are explicitly represented, and (2) the type of model-image correspondences that are made. First, the appearance of the occluding contour, including the formation and persistence of T-junctions, is represented. There is strong information available in the occluding contour, but the difficulty in adequately and explicitly describing it has prevented its use in the past. Second, the shape and topology of the occluding contour is usually stable over a range of viewpoints but is not generated by a fixed, intrinsic feature of the shape. For example, a T-junction produced by the projection of a smooth shape persists over an open set in the space of viewpoints, yet the 3D points that project to the T-junction are not fixed over that set. Consequently, a correspondence between a T-junction detected in an image and a model T-junction is different from a point-point correspondence; a T-junction correspondence defines a connected set of viewpoints where the T-junction can occur rather than a single transformation to bring the model features into exact correspondence with the image. This notion of correspondence is more qualitative, producing a small, constrained region of viewpoints rather than a single viewpoint that generates the image features. This implies a two-step procedure for viewpoint determination: (1) finding a constrained region of viewpoints, and (2) find a single viewpoint within the region as a solution.

This thesis is organized as a progressive study of the viewer-centered representation of 3D shape and its application to the computer graphics problem of interactive viewing and model-based computer vision problem of viewpoint determination. Chapter 2 reviews some of the current work that is closely related to these problems. This review is not exhaustive, but serves to define terms and set the context for the contributions made by this work.

Chapter 3 develops the viewer-centered appearance model for polyhedra, and presents the application of this model to the problem of the interactive display of static polyhedral scenes. The chapter is divided into five main parts: representational issues, algorithms, analysis and results, hybrid methods, and promising future work. The representational issues include the details of the construction and complexity of the data for display. An empirical analysis of display rates and times supports our claim that certain classes of scenes can be displayed more efficiently using this viewer-centered representation than with other known methods. The current and future work section extends and relaxes some of the current constraints.

Chapter 4 describes the rim appearance representation, a viewer-centered representation of the occluding contour of polyhedra. The geometry of self-occlusion for polyhedral edges is described, and an algorithm based on this geometry and on visual events is given for computing the visual event data for the occluding contour edges. The details of this representation and its multi-level development in Chapter 5 is one of the major contributions of this thesis.

Chapter 5 describes an intermediate-level description of the behavior of the occluding contour over viewpoint. This description is based on the lower-level visual event data computed in the rim appearance representation. An algorithm for determining viewpoint given a set of occluding contours is described. The empirical results from an implemented version of the approach support the claim that the model-based precomputed appearance efficiently constrains viewpoint and provides strong constraints for solving for viewpoint and for model-based recognition.

Chapter 6 summarizes the main results of the thesis and comments on several current research directions.

Chapter 2

Related Work

A viewer-centered representation is one that encodes information about the way a shape appears from the point of view of the observer. This is different from an object-centered model in that the viewer-centered model makes explicit in advance the appearance properties of the projected model rather than supplying an algorithmic method for generating the appearance without computing or storing it in advance. The primary use of the object-centered model has been as a verification tool. That is, the hypothesized appearance of an object-centered model can be computed when necessary using the appropriate algorithm, although the appearance is only computed after an hypothesis is made. In contrast, the viewer-centered model computes and stores in advance various features of the way a shape appears and these features are used in the hypothesis formation process itself.

The two main problems that are addressed in this thesis are the interactive display of 3D scenes and the model-based determination of pose from the occluding contour. These problems are approached using a viewer-centered representation of shape based on the definition of the visual event. Section 2.1 briefly reviews work related to the visual-event-based representation described in this thesis. Section 2.2 discusses selected work related to interactive display, and Section 2.3 briefly discusses previous and current approaches to model-based pose determination. The related problem of detecting features of the occluding contour is itself an active research problem. Section 2.4 mentions the current work on detection the occluding contour.

2.1. The Viewer-Centered Model

A common goal of viewer-centered representations is to exploit the regularities in the 2D views of a given 3D model. Because is is impossible to store all possible views of a 3D object, viewer-centered representations must exploit regularities in sets of views for a single object. Exactly how to do this is a difficult problem since a 3D object can appear very differently from different views. Researchers have divided the views of a 3D object in two different ways: (1) the uniform division of the space of all viewpoints [Goad83, Ikeu87, Korn87, Feke84]; and (2) the division of the space of viewpoints into sets based on some definition of equivalence [Bowy89, Egge89, Plan90, Gigu90, Krie89, Srip89].

Although the uniformly-divided viewpoint space is often a fair approximation of the appearance of an object, several problems remain. First, the appearance of an object from a single viewpoint is taken to be representative of an entire region of viewpoints. This is necessary in order to discretize the continuous set of viewpoints. For example, Ikeuchi [Ikeu87] divides the view sphere into 240 triangles. The appearance of a 3D object from the center of a single triangle is taken to be representative of the appearance over the entire triangular patch. The combination of a patch that contains large changes in the appearance of a 3D object and a patch that is too coarse can yield an unacceptable representation of the appearance. Second, these multi-view representations treat individual 2D views independently of each other. In truth, the appearance of 3D shape changes smoothly with viewpoint. There is information about shape and viewpoint in the way shape *changes*. This dynamic quality is not captured in multi-view representations. Finally, the space requirement for a single model at the necessary resolution can become too large to be practical in many situations.

The aspect graph approach in part alleviates the approximation problem of the multi-view representation. The aspect graph is a description of the topological changes in the appearance of a 3D object. Topological changes are defined by the structure of the 2D singularities in the projection of the object. The topological changes in the appearance of a 3D object are used to induce a division of viewpoint space and thus to construct the aspect graph representation.

There are well-known difficulties with the aspect graph scheme. First of all, the detection of features such as faces is very difficult and not robust. The segmentation of faces detected in the image into those belonging to a single object is equally difficult. The indexing problem of selecting the appropriate aspect from the graph is difficult because of the size and the lack of organization of the graph. Another problem is the size of the aspect graph for even very simple polyhedra [Plan88]. The extension of the method to large databases of models seems infeasible without significant restructuring. Finally, the cost of the computation of the aspect graph is large.

The viewer-centered representation presented in this thesis focuses on individual features rather than a constant global topology. The shift from the topology-based focus of the aspect graph to a representation of the geometry of features as a function of viewpoint is the significant new contribution in our approach. The feature representation also improves on some of the size problems as well as the problem of indexing.

2.2. Interactive Viewing

There is a tradeoff in solving the problem of interactive viewing between off-line and on-line solutions. At one extreme is complete off-line image rendering [Denb86]. The on-line phase is a fixed playback of precomputed images that have been rendered with raytracing or other off-line techniques [Glas88, Cook84]. The other extreme is to use standard display algorithms such as Z-buffering to render each frame of the sequence independently. The main drawbacks of the total precomputation approach are the size and inflexibility of the resulting animation description. Rendering each frame at display-time, however, requires too much computation to be interactive.

There are several approaches to this problem that use intermediate representations to reduce the work at display time. Hubschman and Zucker introduced the idea of using frame-to-frame coherence to decrease the time required for hidden-line removal [Hubs81]. This work was not extended to any objects other than convex polyhedra. Shelley and Greenberg used frame-to-frame coherence for the generation of an animation sequence corresponding to a smooth viewpath through a 3D environment [Shel82]. Although the viewpath can be specified interactively, the computation of the appearance of the scene along the viewpath is done off-line and then displayed.

The Binary Space Partition Tree (BSP-tree) has been used to display polyhedral scenes in near real-time by precomputing a structure that gives relative depth-ordering for faces in the model [Fuch83, Nayl90]. The BSP-tree does not take advantage of the frame-to-frame coherence in animation sequences, however. In addition, the BSP-tree approach only applies to hidden-surface removal, and requires drawing all of the polygons in the BSP-tree, which is often more than the number of polygons in the scene since polygons may have to be split in constructing the BSP-tree.

In contrast to these previous approaches, our algorithm is based on visual-event computation and places no restrictions on the model polyhedra. Arbitrary nonconvexities in the models are allowed, and the hidden-line algorithm extends to the hidden-surface case. The appearance of the scene is generated on-line as the viewpath is interactively defined. In cases where hidden-line removal is a desirable or acceptable alternative to raster display, our approach for hidden-line animation may be used to achieve greater frame rates. Our algorithm only draws the visible polygons, usually much less than the total number of polygons in the scene.

2.3. Model-Based Object Recognition and Pose Determination

Approaches to the viewpoint determination problem must solve two inter-related sub-problems: finding the correct correspondence of model and image features, and recovering the viewpoint that maximizes the match of corresponding features [Grim90]. Solutions vary from solving these two problems independently to treating them as a combined process, with the goal of solving for viewpoint. For example, iterative methods assume that the model-image correspondence is known, and solve for viewpoint by applying numerical techniques to revise an initial estimation of viewpoint [Lowe87, Ponc89, Worr89]. Similarly, the alignment approach [Basr88, Hutt90] and parameter space methods [Thom87] assume a model-image correspondence in order to derive a unique viewpoint. Interpretation tree methods solve for correspondence and viewpoint simultaneously by using a constrained search through a tree of model-image correspondences [Grim90a]. Finally, viewer-centered representations such as characteristic views [Wang90] and the aspect graph [Plan87, Bowy89, Gigu90] precompute representative sets of viewpoints and then attempt to solve for the best correspondence between image features and the features in each representative view.

Our approach is to use the appearance of the occluding contour to solve for a set of viewpoints that can globally constrain where a single viewpoint solution must lie. In the case of iterative methods [Lowe87], a starting viewpoint can be obtained from this small viewpoint set. Parameter space methods [Thom87] that can avoid a costly search of the entire space of transformations become much more efficient when they need only consider this reduced set of viewpoints. The search of an interpretation tree [Grim90] is also made more efficient in space and time when there are global constraints on the possible solutions. Aspect graph methods [Bowy89, Krie90, Ikeu87] must address the problem of how to select a few aspects to test from a large number of potential aspects.

2.4. Detecting Occlusion Features

Edges in an image that are part of the occluding contour are produced at depth discontinuity boundaries in the scene. Standard edge detectors that are based on static image intensities alone cannot reliably distinguish between occluding contours and other types of scene edges [Barr81]. With additional information, such as depth data from rangefinders or from stereo methods, and dynamic data from motion, occluding contours can be segmented and analyzed. In addition to the segmentation issue is the task of detecting features of the occluding contour such as curvature extrema and T-junctions.

Work on detecting the occluding contour from optic flow has focused on the occlusion boundary [Thom85], where optic flow constraints are violated. Discontinuities in the flow field are interpreted as depth discontinuities in the scene. These methods depend on the dynamic flow data available from a moving viewer or a dynamic scene.

Recent work on detecting occlusion boundaries using stereo shows promising results [Toh90]. Stereo methods provide depth data that are used in conjunction with image intensities to segment those image edges that lie on depth discontinuities. The robustness of the stereo algorithm is central to this approach. Unfortunately, most stereo algorithms lack robustness at occlusion boundaries, Other recent work on surface reconstruction has used stereo to detect and use occluding contours [Vail89] for surface reconstruction. Researchers have also begun to integrate stereo data and dynamic information in order to segment and measure the curvature on occlusion boundaries [Cipo90]. Contours are tracked throughout a dynamic image sequence using an energy-based contour tracking method[Kass88] in order to model how the contour changes as viewpoint changes. This information is used to reconstruct the properties of the occluding contour and the 3D surface generating it.

Spatiotemporal representations are formed from images over time that are stacked together into a cube of data. There are characteristic features in the spatiotemporal cube that correspond to occlusion in an image sequence [Bake88, Allm91]. These occlusion features, such as curves flowing through the spatiotemporal cube that merge and split, are part of the occluding contour and correspond to depth discontinuities in the scene.

Chapter 3

Interactive Viewing

Interactively viewing a static scene requires the computation and display of a sequence of images generated from a user-controlled, continuously-changing viewpoint. The goal of a system for interactive viewing is to display a 3D scene as it would appear from a particular vantage point, and to update the appearance of the scene dynamically as the user interactively modifies the vantage point within the environment. There must be a balance, however, between the incompatible goals of realism and speed in the interactive display. Clearly it is important to display the scene as fast as possible to give the user the illusion of movement within the environment.

Computing and displaying such animation sequences has many applications. For example, computer-aided design of 3D objects requires the interactive construction and display of objects from a wide range of viewing directions . The animated display of a rotating 3D object gives the user a sense of depth and structure that is useful in design and visualization [Farr85]. The animated presentation of a scene that can be navigated dynamically has application in flight simulation [Yan85] and architectural walk-through [Broo86].

There are two essential requirements for this type of animated display: realism in each image and video-rate display. Without an appropriate level of realism, the effect of interaction with an environment is lost. Perceptual continuity is lost when the display rate is too slow. In meeting the requirements of realism and video-rate display there is a fundamental trade-off between off-line and on-line solutions. The precomputation of information can be as extreme as complete image rendering off-line. The on-line animation phase is then reduced to a playback of the precomputed images [Denb86]. There are ways to compute realistic images off-line using, for example, ray tracing techniques [Glas88, Cook84], but the high per-frame cost makes interactive animated display using these approaches intractable. The main drawback of the total precomputation approach is the size and inflexibility of the resulting animation description. At the other end of the spectrum is the synthesis of the frames of the animation sequence on-line. Animation in this case depends on fast dynamic frame display [Fuch83]. Without some form of precomputation, however, video-rate display and realistic image synthesis become almost impossible. Wire-frame display is typical for low-cost video-rate display without significant precomputation.

This chapter discusses the problem of how to display a polyhedral scene interactively from a moving viewpoint. This will be referred to as *interactive viewing*; the restricted viewer motion of a great circle on the unit sphere under orthographic projection is considered in detail. A polyhedral scene can be rendered with varying degrees of detail, ranging from wire-frame display to visible-surface display using a shading model. The basic algorithm first introduced by Plantinga [Plan88] is presented for animating the display of a polyhedral scene with hidden lines removed; this algorithm is modified to allow display with hidden surfaces removed, including the use of shading models, multiple light sources and shadow computation. For the hidden-line computation, the results presented here show that it possible to display the sequence at a rate comparable to the rate of display of wire-frame models without hidden-lines removed.

The algorithm is based on the the exploitation of *viewpath* coherence, a form of frame-to-frame coherence, by computing the appearance of the scene in the first frame, and then computing the viewpoints on the viewpath at which the structure of the scene changes qualitatively. Qualitative changes are changes in the structure of the projected scene. The viewpoints where structural changes occur are computed through the construction of the *asp* for the scene, a representation that makes explicit exactly which vertices, edges, and faces are visible as a function of viewpoint. The algorithm has two phases: a preprocessing phase, in which the initial appearance of the polyhedron and the events are computed and the visible edge graph is constructed; and an on-line, interactive phase, in which a sequence of frames along a user-controlled viewpath is displayed in real time.

The asp quantifies how all possible kinds of *visual events* occur as a continuous function of viewpoint. Informally, a visual event can be thought of as any change in the scene as a result of occlusion. Thus the disappearance of a face as it turns away from the viewing direction is one such event. The partial occlusion of an edge by some other edge (the beginning or ending of a T-junction) is another kind of event. More formally, all structural changes in the projection of a polyhedral scene are the result of the overlap in the image plane of three edges [Plan89, Gigu90]. In the case of a face that turns away from the viewing direction, the point at which the face is edge-on to the viewer is a degenerate form of the apparent intersection of three edges. The occlusion of an edge by another face creates a T-junction that begins and ends at two distinct viewpoints along a viewpath. The T-junction always begins and ends at a viewing direction where an edge and a vertex appear to intersect. This event (EV-event) is another form of the apparent intersection of three edges where two of the edges meet at a vertex and actually do intersect.

The orthographic viewing model can be characterized as the set of viewpoints on the surface of the unit sphere $\mathbf{S}^2$. Any viewer movement can be described as a 1D path of viewpoints on the surface of $\mathbf{S}^2$. In general, the apparent intersection of three edges (EEE-event) occurs at no more than two points along a 1D path of viewpoints on $\mathbf{S}^2$, and corresponds to a topological change in the appearance of the scene. The location of all such events can be computed and ordered sequentially along the 1D viewpath. By precomputing and ordering the viewpoints where topological changes occur, the coherency between viewpoints is exploited. This coherence between frames is a result of the fact that for most small changes in viewpoint along a smooth viewpath, only linear changes in the appearance of the scene take place [Ikeu87]. Linear changes are those changes in the viewpoint that do not change the structure of the projected line drawing in the image. Changes in the structure, i.e., topological changes in the image structure graph (ISG), are characterized in the asp as visual events corresponding to the apparent intersection of edges in the image. In order to take advantage of the coherence between viewpoints and hence between frames, these topological changes are explicitly computed and stored. These events are an exact representation of the changes in the appearance of the ISG. At runtime, a viewpath through the space of viewpoints determines those events that are relevant to updating each successive frame in an animation sequence.

Hubschman and Zucker introduced the idea of using frame-to-frame coherence to decrease the time required for hidden-line removal [Hubs81]. They worked in a world with a small number of stationary convex polyhedra and they found a number of frame-to-frame coherence constraints. The result was a partition of the scene such that "the movement of the viewing position across a partition boundary results in an occlusion relationship becoming active or inactive." The scene is updated when one of these boundaries is crossed. As a result, the storage requirements are $\Theta(\mathbf{n}^3)$ even in the case of a single, nondegenerate convex polyhedron. A generalization of this technique to multiple non-convex polyhedra would result in worst-case storage requirements of $\Theta(\mathbf{n}^9)$ for a scene with **n** faces [Plan87]. The algorithm presented here places no restrictions on the model polyhedra, allowing arbitrary nonconvexities, and extends to the hidden-surface case.

Shelley and Greenberg used frame-to-frame coherence for the generation of an animation sequence corresponding to a smooth viewpath through a 3D environment [Shel82]. A smooth viewpath is represented as a B-spline, which can be interactively defined. Path coherence (frame-to-frame coherence along a single viewpath) is exploited to reduce the expense of the sorting and culling operations for visible line and visible surface computation. Rendering makes use of a priority ordering of polygons in the environment, and changes in this ordering are computed when the B-spline viewpath crosses one of the separating planes between objects in the environment. Although the viewpath can be specified interactively, the computation of the appearance of the scene along the viewpath is done off-line and then displayed. The approach reported here generates the appearance of the scene on-line as the viewpath is interactively defined.

The Binary Space Partition Tree (BSP-tree) has been used to display polyhedral scenes in near real-time by precomputing a structure that gives relative depth-ordering for faces in the model [Fuch83, Nayl90]. The BSP-tree simplifies the hidden-surface computation for an arbitrary viewing direction by encoding the relative depth ordering of the model implicitly in a tree structure for all viewing directions. The display of a single frame from some viewpoint with hidden-surfaces removed involves traversing the BSP-tree to generate a list of polygons in back-to-front order. The BSP-tree does not take advantage of the frame-to-frame coherence in animation sequences, however. Each frame of a viewpath is generated by a separate and complete traversal of the BSP-tree structure. Also, the BSP-tree approach only applies to hidden-surface removal. In cases where hidden-line removal is a desirable or acceptable alternative to raster display, our approach for hidden-line animation may be used to achieve greater frame rates. Finally, the BSP-tree approach requires drawing all of the polygons in the BSP-tree, which is often more than the number of polygons in the scene since polygons may have to be split in constructing the BSP-tree. Our algorithm only draws the visible polygons, usually less than the total number of polygons in the scene.

The rest of this chapter focuses on how to efficiently apply the computation of visual events in a space of viewpoints to the problem of interactive viewing with either the hidden lines or hidden surfaces removed. The main previous results related to the asp [Plan88] are reviewed in Section 3.1. Section 3.1 also defines the orthographic viewing model and the visual event. Section 3.2 presents a detailed discussion of the computation and organization of visual events for display. First the hidden-line computation is discussed, followed by the problem of hidden-surface computation, including issues involving shaded display and an efficient, object space approach to finding shadow regions. Section 3.3 reports the computational results from an implementation of the hidden-line method. Hybrid approaches are discussed in Section 3.4, combining the strengths of the visual-event based approach with other methods in order to solve some of the weaknesses of the visual-event based method. Event-based depth ordering reduces the visual event complexity by relying on depth ordering to remove the hidden surfaces. Model-space hierarchies organize the scene into components so that smaller pieces of the scene are insulated from changes. This helps to avoid a large, repeated precomputation cost for small changes in the scene structure. Using the occluding contour to approximate the shape to be displayed reduces the amount of visual event information and the model construction cost. Section 3.5 concludes with a review of the contributions of this work and the promising future directions.

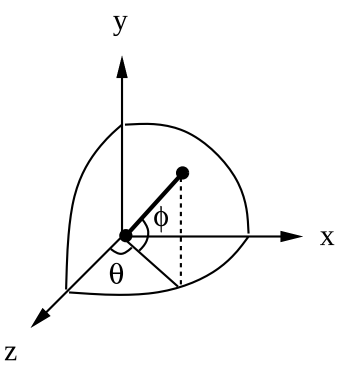


Figure 3.1. The orthographic viewing model is defined as the set of unit vectors on the view sphere.

3.1. The Asp

The asp [Plan88, Plan87, Plan86] is a representation that quantifies the visual events in a 3D polyhedral model that arise as a result of occlusion. In general, the visual events that are explicitly represented in the asp are the apparent intersections of edges. Under the orthographic projection model, the apparent intersection of two edges (a T-junction) corresponds to a 2D surface in aspect space. Similarly, the apparent intersection of three edges corresponds to a 1D curve, and four edges that appear to intersect form a single point in aspect space. An actual vertex in the scene can be thought of as a degenerate apparent intersection, where the edges actually do meet at the vertex. Since these various visual events are explicit in the asp, the appearance of an edge from a given viewing direction can be easily computed. The exact appearance of the edge is bounded by the visual events involving other occluding edges in the scene.

The viewpoints where visual events occur depend on both the space of viewpoints and the models being displayed. The following subsections define the viewing model, the object model, and formulate the general idea of the visual event. Aspect space and the asp are then presented in detail as a tool for computing the visual events for polyhedra under the orthographic viewing model.

3.1.1. The Object and Viewing Models

The orthographic viewing model is the parallel projection of an object onto the image plane. Figure 3.1 shows the projection model as a unit sphere, with the directions on the sphere determined by the angles θ and ϕ. A projection from a direction (θ, ϕ) is created with this model by rotating the scene by (θ, ϕ) and then projecting into the $x-y$ plane. Thus, a point (x_0, y_0, z_0) in $\mathbb{R}^3$ to be projected from the direction (θ, ϕ) undergoes the following transformation:

$$\begin{bmatrix} x_0, y_0, z_0 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{3.1}$$

so that the coordinates (u, v) of the projected point in the image plane become

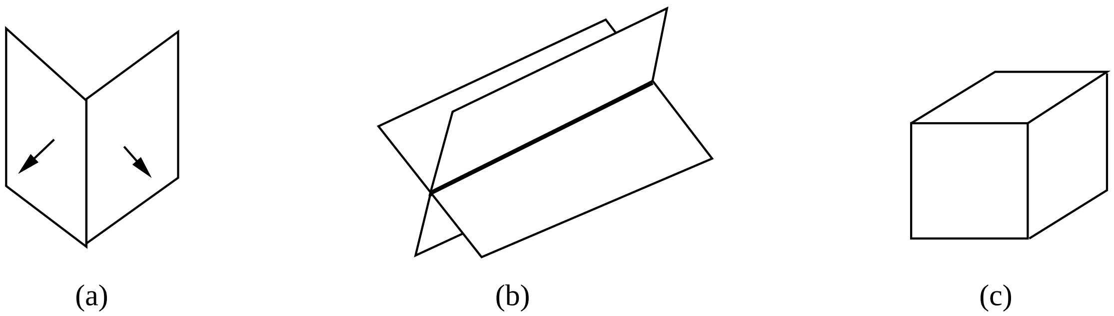


Figure 3.2. (a) Each face of the polyhedron is oriented, and each edge is formed by exactly two faces. (b) An example violating the assumptions where the central edge is formed by four faces rather than two. (c) The polyhedral object model must form a closed volume.

$$
\begin{aligned}
u &= x_0 \cos\theta - z_0 \sin\theta \\
v &= x_0 \sin\theta \sin\phi + y_0 \cos\phi + z_0 \cos\theta \sin\phi
\end{aligned}
\tag{3.2}
$$

The object model that is used to represent 3D objects is the polyhedron. It is assumed that there is a directed normal associated with each face of the polyhedron. A face is said to be visible when the dot product between the viewing direction **V** and the normal to the face **N** is positive:

$$\text{Face } \mathbf{f} \text{ is visible} \Leftrightarrow \mathbf{V} \cdot \mathbf{N} > 0$$

Faces in the polyhedron meet at edges, and it is assumed that exactly two faces meet at an edge. As shown in Figure 3.2, faces are oriented, each edge is a result of exactly two intersecting faces, and the faces must form a closed volume. These restrictions can be relaxed and handled as special cases, although for simplicity they are enforced in the work described here.

3.1.2. Appearance, Visual Events, and Image Structure

The *appearance* of a 3D model is a description of the geometry of the visible portions of that model in the 2D image plane. Appearance is a function of the geometry of the 3D model, the viewpoint, and the projection mapping. Certain singularities can occur under projection from $\mathbb{R}^3$ into the 2D image plane, and these singularities have been termed *visual events*. The singularities of the projection mapping have been studied by Whitney [Whit55], and there is work being done toward understanding the form of singularities under projection of specific 3D models such as polyhedra [Plan90, Gigu90, Stew88], surfaces of revolution [Krie89, Egge89], parametric surfaces [Srip89, Ponc90], and generic surfaces [Koen76, Rieg87].

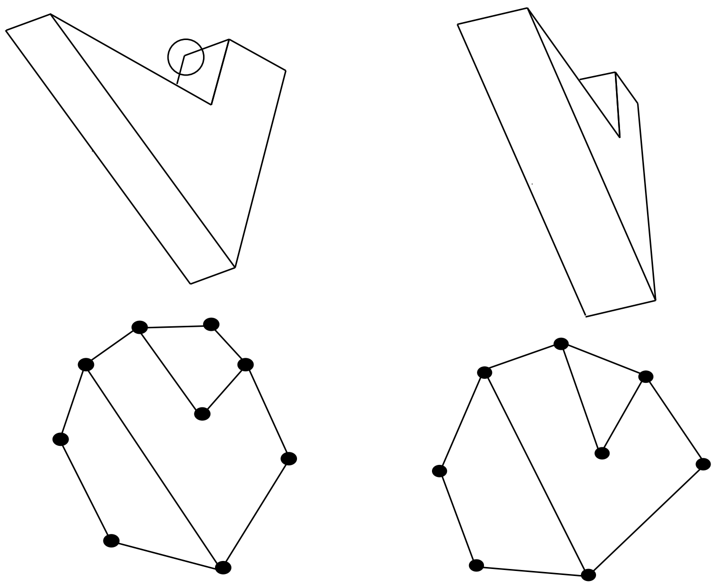


Figure 3.3. The image structure graph (ISG) is the graph corresponding to the line drawing of a projected polyhedron.

The singularities (visual events) for polyhedra can be completely characterized as the intersection in the image plane of sets of non-adjacent model edges [Plan90, Gigu90]. These events and their inter-relationships form the *image structure graph* (ISG), a graph where each node is a vertex and each incident edge is a projected model edge. The appearance of a polyhedron, commonly known as a line drawing, forms an ISG where each vertex and T-junction is a node, and each projected polyhedral edge is a graph edge. Figure 3.3 shows two projections of a polyhedron and their associated ISGs.

An alternative (and slightly different) definition of the visual event is any type of event that causes a topological (structural) change in the ISG. The stability of the event is the difference in the two definitions. A singularity in the projection map may be stable for an open set of viewpoints in viewpoint space. A visual event causing a topological change in the ISG is not stable, and persists for only a degenerate set of viewpoints. For example, a T-junction does not always cause a structural change in the ISG, although a T-junction is a singularity in the projection map. The birth and death of a T-junction, however, causes a topological change in the ISG, and in this sense are considered visual events. Figure 3.3 shows the ISG for two different but closely-spaced views of a polyhedron. The disappearance of the circled vertex causes the structural change in the ISG, and is the birth of a T-junction. Thus the singular view where the structural change occurs is the visual event. The T-junction itself is a visual event in the sense that it is a singularity of the projection map.

We will consider a visual event to be a singularity of the projection mapping. The distinction between the stable T-junction and other nonstable singularities is important, and the geometry of the T-junction will be developed in Chapters 4 and 5.

3.1.3. Aspect Space

The appearance of a polyhedron is a 2D set of line segments that is generated from the 3D polyhedral description under a specific projection mapping for a single viewpoint. The central characteristic of the asp for a polyhedron is that the asp represents the appearance of the polyhedral faces for *all* viewpoints rather than a single viewpoint [Plan88]. Specifically, the asp for a single face is a volume in the 4D space of image space $\times$ viewpoint space, i.e., in $\mathbb{R}^2 \times \mathbf{S}^2$. The boundaries of this volume are hypersurfaces that correspond to the visibility of the edges bounding the face. The edge hypersurfaces are bounded by surfaces that correspond to the visibility of the vertices of the face. The volume in aspect space that corresponds to the visibility of a face is not a polytope since the surfaces are not planar. The equations for the surfaces are well-behaved, however, and are algebraic ruled surfaces of degree at most three. Therefore the asp for a face can be represented and manipulated in much the same way as a polytope. For example, the intersection of two volumes in aspect space can be computed exactly in closed form.

The asp was initially studied in order to construct the aspect graph. The work by Plantinga [Plan88] includes a complete discussion of the definition, construction, and properties of the asp. As Plantinga describes, a fundamental property of aspect space is that occlusion in object space corresponds to set subtraction in aspect space. Thus, the visibility of a face from all viewing directions can be computed exactly by performing set subtraction in aspect space. The process of set subtraction in aspect space generates explicit equations for the 2D surfaces, 1D curves, and vertices mentioned above.

The restricted problem of constructing the asp for one degree of freedom in viewpoint simplifies the form of the asp because the allowable viewpoints are constrained to lie on a great circle (i.e., $\phi = 0$) of the view sphere (after an appropriate coordinate transform). By making this simplification, aspect space is reduced in complexity from four degrees of freedom to three. The asp for a polygonal face then becomes a 3D volume bounded by the visibility of its edges. The visibility of an edge becomes a surface in the 3D aspect space. All of the visual events are simplified by this viewpoint constraint, so that the apparent intersection of two edges becomes a curve of viewpoints, and is bounded by the apparent intersection of three edges, which is a point in aspect space. The equations that generate these volumes, surfaces, curves and points are obtained directly from the general form of the asp by restricting the viewpoints to lie in a plane. This is equivalently done by setting one component of the viewing direction to 0.

The algorithm for constructing the asp for a face **F** is to subtract from the asp for **F** the asps for the faces in front of **F**. Subtraction is set subtraction, and faces in front of **F** are those faces that occlude **F** from some viewpoint. This algorithm hinges on the observation that the occlusion of **F** by other faces in the scene corresponds to the subtraction of the asps for the occluding faces from the asp for **F**. When the viewpoint is constrained to the equator of the view sphere, the subtraction of one asp from another corresponds to the subtraction of the region of intersection of two 3D volumes.

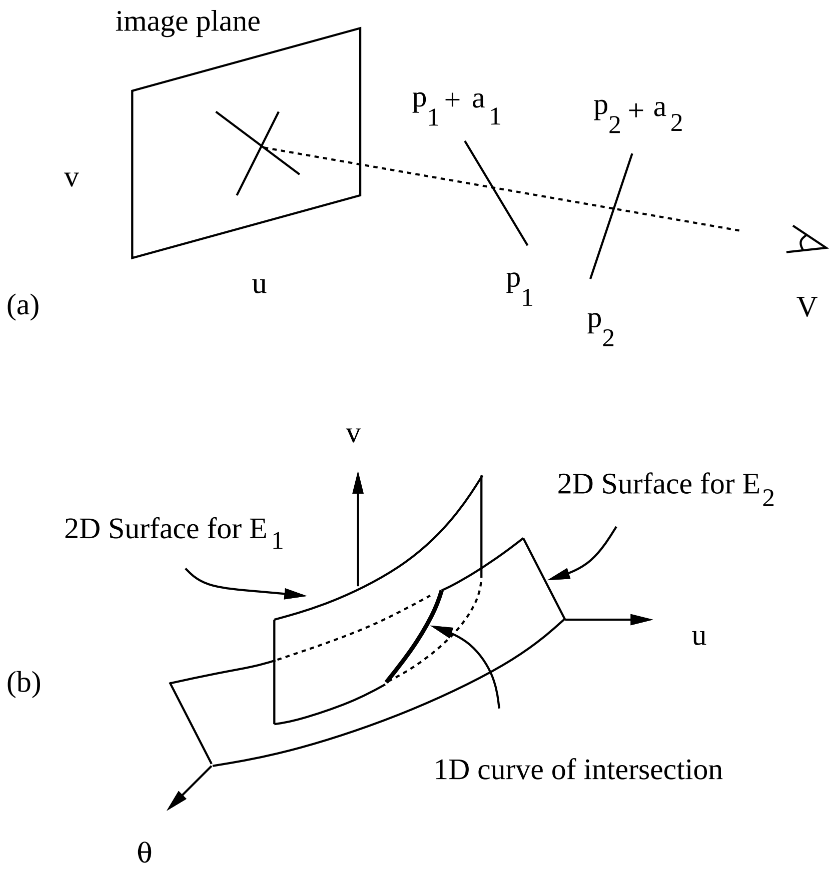


Figure 3.4. The apparent intersection of two edges in the image plane.

Specifically, the construction of the asp for a scene is done by defining the intersection of the boundaries of the 3D volumes in aspect space corresponding to the visibility of faces. The 2D surfaces in aspect space bounding a 3D volume correspond to the visibility of the edges of a face. The intersection of two 2D surfaces in aspect space is a 1D curve that lies on each of the 2D surfaces. Since each 2D surface in aspect space corresponds to the visibility of an edge in the model, this 1D curve represents the apparent intersection of the two edges in the image. For a specific value of the viewpoint parameter θ, the equation of the 1D curve that lies on the 2D surface can be used to compute the exact image coordinates of the apparent intersection point of the corresponding two edges.

The visibility of an edge is bounded by the other edges in the scene that occlude it from various viewing directions. The construction of the asp for an edge can be viewed as the computation of the equations of all of the 1D curves that lie on the 2D surface for that edge.

3.1.4. Analysis and Equations

Computing all curves for a 2D surface in aspect space involves finding the intersection of the 2D surface with all other 2D surfaces that correspond to the other edges in the scene. Consider, for example, two edges $\mathbf{E}_1$ and $\mathbf{E}_2$ in $\mathbb{R}^3$. Let $\mathbf{E}_1$ be an edge from point $\mathbf{p}_1 = (x_1, y_1, z_1)$ to point $\mathbf{p}_1 + \mathbf{a}_1$ where $\mathbf{a}_1 = (a_1, b_1, c_1)$. Let $\mathbf{E}_2$ be an edge from point $\mathbf{p}_2 = (x_2, y_2, z_2)$ to point $\mathbf{p}_2 + \mathbf{a}_2$ where $\mathbf{a}_2 = (a_2, b_2, c_2)$. The edge $\mathbf{E}_1$ can be parametrized in $\mathbb{R}^3$ by the expression $\mathbf{p}_1 + s\ \mathbf{a}_1$, where $0 \le s \le 1$. The viewing direction $\mathbf{V}$ can be described in spherical coordinates as the vector $\mathbf{V} = (\sin\theta,\ 0,\ \cos\theta)$ that depends on the single viewpoint parameter θ. The equation of the image point where two edges $\mathbf{E}_1$ and $\mathbf{E}_2$ appear to intersect in the image plane (u,v) is a function of the viewpoint parameter θ and is given by [Plan88]:

$$u = (x_1 + s\ a_1)\cos\theta - (c_1 + s\ z_1)\sin\theta \qquad (3.3)$$
$$v = y_1 + s\ b_1$$

where the parameter s is expressed as

$$s = \frac{\mathbf{V} \cdot ((\mathbf{p}_2 - \mathbf{p}_1) \times \mathbf{a}_2)}{\mathbf{V} \cdot (\mathbf{a}_1 \times \mathbf{a}_2)} \qquad (3.4)$$

Figure 3.4 shows the edges $\mathbf{E}_1$ and $\mathbf{E}_2$ in $\mathbb{R}^3$ and the apparent intersection of $\mathbf{E}_1$ and $\mathbf{E}_2$ in the image plane. Given a fixed value of the viewpoint parameter θ, Equation 3.3 gives the image coordinates of the apparent intersection of the edges $\mathbf{E}_1$ and $\mathbf{E}_2$.

The 1D curve described by Equation 3.3 corresponds geometrically to the intersection in aspect space of the 2D surfaces generated by edges $\mathbf{E}_1$ and $\mathbf{E}_2$. The 1D curve of intersection extends over some range of viewpoints $[\theta_i, \theta_j]$. Figure 3.4(a) shows the edges $\mathbf{E}_1$ and $\mathbf{E}_2$ as they would appear in the image plane from a single viewpoint. As the viewpoint moves along the rotation path, the intersection of the edges in the image changes. Figure 3.4(b) shows the 2D surfaces in aspect space corresponding to $\mathbf{E}_1$ and $\mathbf{E}_2$. The 1D curve on this surface is the trace of the apparent intersection of $\mathbf{E}_1$ and $\mathbf{E}_2$. Figure 3.4(b) illustrates how the 1D curve that is the trace of the apparent intersection of $\mathbf{E}_1$ and $\mathbf{E}_2$ corresponds to the intersection of two 2D surfaces in aspect space. This 1D curve is described by Equation 3.3. The appearance of the two edges in the image plane is a 2D cross-section for a fixed value of the viewpoint parameter θ.

The most general visual event is the apparent intersection of three edges, i.e., the EEE-event. This visual event corresponds to the intersection on a 2D surface of two distinct 1D curves. Computing the viewpoints where EEE-events occur involves finding the intersection of 1D curves that were generated by edges that do not lie on the same face. The viewpoint of intersection where two curves on the same 2D surface coincide can be found directly by solving the equation for viewpoint given the equations of the curves and the 2D surface on which they lie. Consider three edges $\mathbf{E}_1$, $\mathbf{E}_2$, and $\mathbf{E}_3$ that lie in $\mathbb{R}^3$ as illustrated in Figure 3.5(a). $\mathbf{E}_1$ connects the points $\mathbf{p}_1$ and $\mathbf{p}_1 + \mathbf{a}_1$. Likewise, $\mathbf{E}_2$ connects the points $\mathbf{p}_2$ and $\mathbf{p}_2 + \mathbf{a}_2$, and $\mathbf{E}_3$ connects the points $\mathbf{p}_3$ and $\mathbf{p}_3 + \mathbf{a}_3$. Let $\mathbf{S}_i$ represent the 2D surface in aspect space corresponding to $\mathbf{E}_i$. The intersection of $\mathbf{S}_1$ and $\mathbf{S}_2$ in aspect space is a 1D curve $\mathbf{C}_1$ on $\mathbf{S}_1$. Similarly, the intersection of $\mathbf{S}_1$ and $\mathbf{S}_3$ in aspect space is a 1D curve $\mathbf{C}_2$ on $\mathbf{S}_1$. The intersection of the two curves $\mathbf{C}_1$ and $\mathbf{C}_2$ on $\mathbf{S}_1$ corresponds to the viewing direction where all three edges appear to intersect in the image plane. This viewing direction is equivalent to the direction of some vector through a point $\mathbf{p}_1 + s\mathbf{a}_1$ on $\mathbf{E}_1$, $0 \le s \le 1$, which intersects both $\mathbf{E}_2$ and $\mathbf{E}_3$. A vector parallel to this direction is given by

$$\mathbf{V} = ((\mathbf{p}_1 + s\mathbf{a}_1 - \mathbf{p}_2) \times \mathbf{a}_2) \times ((\mathbf{p}_1 + s\mathbf{a}_1 - \mathbf{p}_3) \times \mathbf{a}_3) \tag{3.5}$$

Since the ϕ-component of the viewing direction is constrained to be 0, Equation 3.5 has the form

$$\alpha s^2 + \beta s + \gamma = 0 \tag{3.6}$$

where α, β, and γ are linear functions of the endpoints of $\mathbf{E}_1$, $\mathbf{E}_2$ and $\mathbf{E}_3$. Equation 3.6 can be solved for the parameter s, and then Equation 3.5 can be solved for the desired

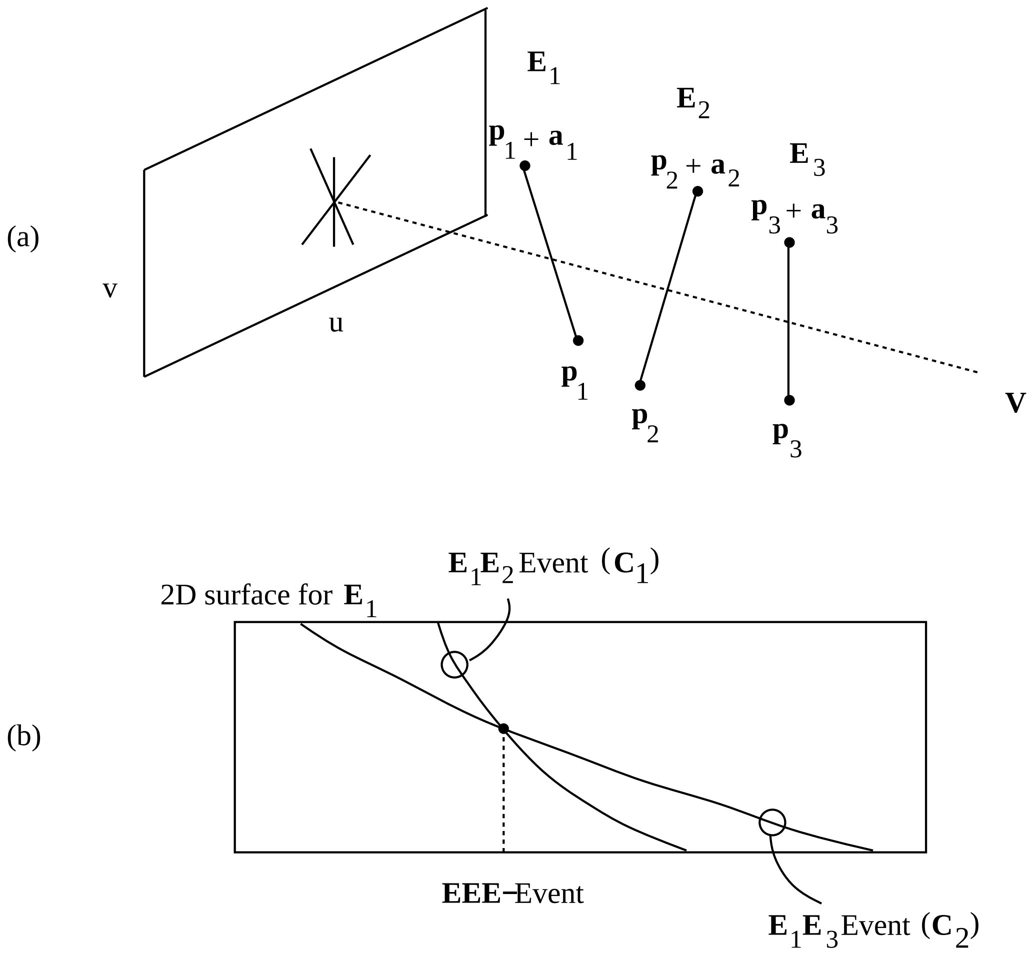


Figure 3.5. (a) The apparent intersection of three scene edges in the image plane. (b) The intersection of two 1D curves on a 2D surface in aspect space

vector **V** [Plan88]. The value of the viewpoint parameter θ for vector **V** is the value of θ where the two 1D curves $\mathbf{C}_1$ and $\mathbf{C}_2$ intersect in aspect space on $\mathbf{S}_1$. This is shown in Figure 3.5(b).

Since the visibility of an edge is a function of the face to which the edge belongs, determining which edges in the scene occlude a given edge can be done quickly using simple tests. For example, edges that have non-overlapping *y*-coordinates in $\mathbb{R}^3$ cannot

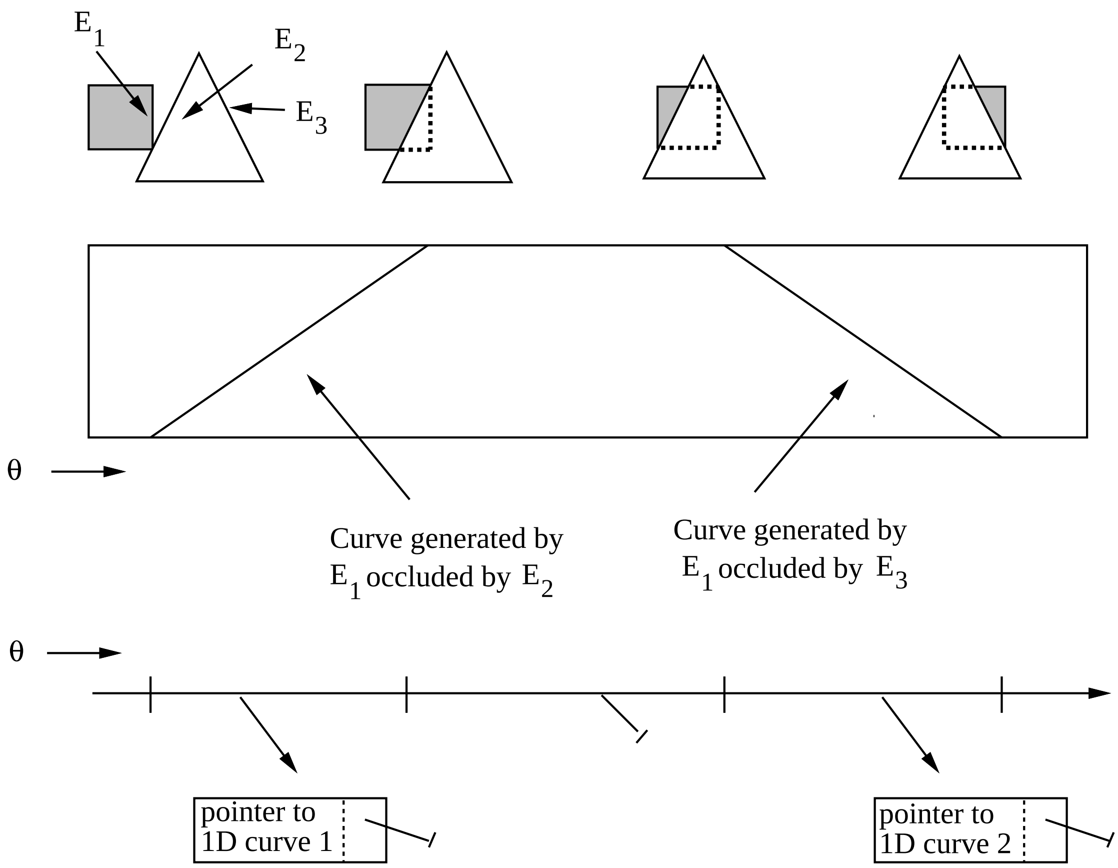


Figure 3.6. The occlusion of an edge of the square by the triangle corresponds to 1D curves on the 2D surface for that edge in aspect space.

occlude each other (remember: $\phi = 0$). The extent of the curve of intersection between two 2D surfaces is bounded by the viewpoint extent of each of the 2D surfaces. In other words, the computed curve of intersection between two 2D surfaces must lie on both 2D surfaces. The viewpoint extent of the curve is therefore determined by the extent of the 2D surfaces in aspect space, which is directly related to the visibility of the corresponding edge as a part of the face to which it belongs.

The purpose for constructing the asp is to compute exactly the viewpoints where visual events occur in the scene, and to use this list of visual events to represent how the scene will appear in the image plane. The complete asp for a polygonal face is the volume bounded in aspect space by the set of 2D surfaces corresponding to the edges of that face. Associated with each 2D surface is the set of 1D curves of intersection generated by all other occluding edges in the scene. Recall that each 1D curve on a 2D surface is the equation of an apparent intersection of two edges in the scene. The visual events of interest are the EEE-events mentioned above: (1) the viewpoints where the edge appears or disappears completely as a part of the face of which it is a part, (2) the viewpoints where a T-junction begins or ends, and (3) the most general EEE-event where three non-intersecting edges appear to intersect at a single viewpoint. The viewpoints that represent these events can be obtained directly from the representation of 1D curves of intersection on a 2D surface. For each edge and its corresponding 2D surface, the viewpoints where the edge appears or disappears completely is the extent in viewpoint space of the 2D surface. This extent is given directly as the visibility of the face to which the edge belongs. The starting and ending of each T-junction for an edge is given by the viewpoint extent of each 1D curve of intersection on the 2D surface for that edge. These are found directly without additional computation, and can be stored in a list of events associated with the 2D surface.

All visual events affecting the appearance of an edge can be derived directly from the 2D surface for that edge and the 1D curves corresponding to the intersection of the 2D surface with all other 2D surfaces in the scene. Once computed, the visual events for each edge can be sorted and stored with the corresponding 2D surface. Figure 3.6 shows the event list for an edge, $\mathbf{E}_1$, of a square that is being occluded by a triangle. The edges of the triangle cause two 1D curves to be generated on the 2D surface for the edge of the square. The viewpoint extent of the 1D curves on the 2D surface are the viewpoints where the T-junctions on $\mathbf{E}_1$ begin and end. The beginning and ending viewpoints correspond to EV-events, and are the visual events in this example that are sorted and stored. As shown in the figure, the appearance of $\mathbf{E}_1$ over some interval $\mathbf{I}$ on the viewpath is described by a list of 1D curves. The set of 1D curves corresponds to T-junctions and hence determines the appearance of the edge over interval $\mathbf{I}$ in the sorted list of viewpoints.

More generally, a list of visual events ε_1, ε_2, ... , ε_k delimits all of the topological changes in the appearance of an edge corresponding to a 2D surface. For each particular

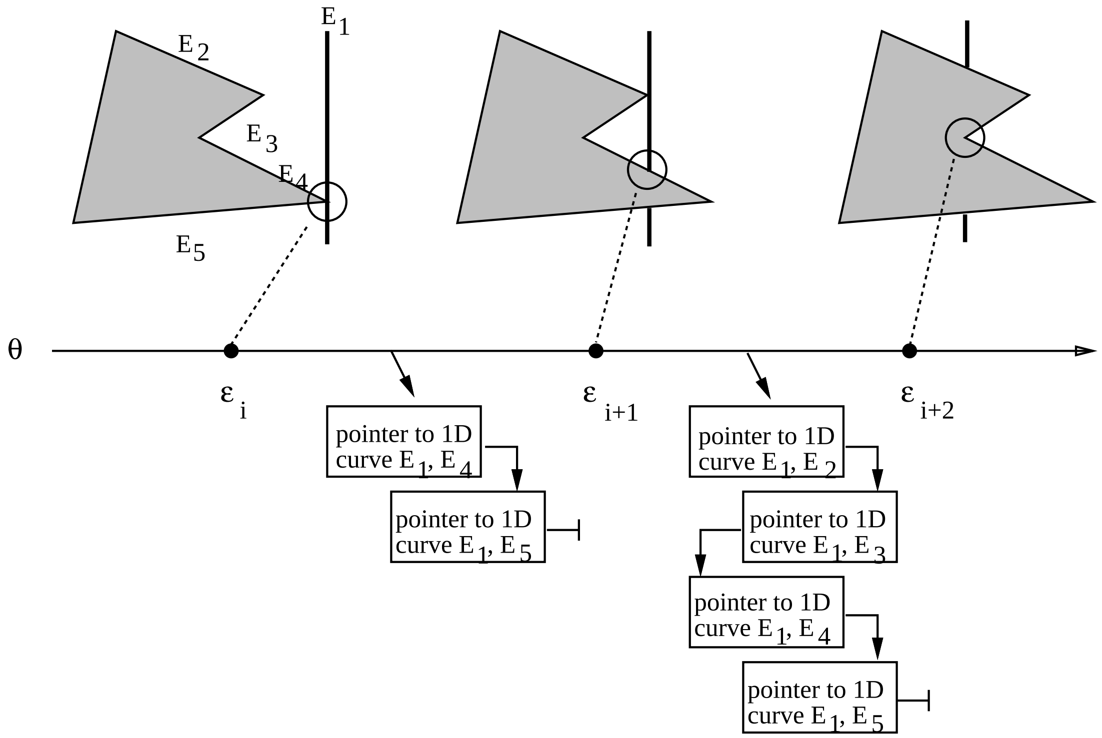


Figure 3.7. The 1D curves in aspect space which determine the appearance of an edge can be ordered for each interval in the list of visual events for the edge.

interval **I** = [ε_i, ε_{i+1}] of the visual event list in the viewpath, the appearance of the edge is completely determined by the set of 1D curves that extend over that interval. Figure 3.7 shows a portion of the visual event list for a single edge $\mathbf{E}_1$ being occluded by a face with five edges. For $\theta = \varepsilon_i$ there is an EV-event where $\mathbf{E}_4$, $\mathbf{E}_5$ and $\mathbf{E}_1$ appear to intersect. At $\theta = \varepsilon_{i+1}$ the EV-event involves $\mathbf{E}_2$, $\mathbf{E}_3$ and $\mathbf{E}_1$. Between viewpoints ε_i and ε_{i+1} the 1D curves in aspect space that determine the visibility of $\mathbf{E}_1$ are the curves generated by $\mathbf{E}_4$ and $\mathbf{E}_5$ with $\mathbf{E}_1$.

All of the visual events affecting the appearance of an edge over an interval in viewpoint space are represented by the set of 1D curves within that interval. Since no visual events occur *within* an interval **I**, the set of 1D curves for an edge can be ordered within **I** based on the orientation of the edge. The 1D curves for interval **I** correspond to T-junctions along the edge. These junctions can be ordered relative to the designated *start* vertex of the directed edge. The ordering is based upon the *a priori* orientation of each edge, i.e., edges are directed from a start vertex to an end vertex, and T-junction locations along the edge can thus be ordered relative to one of these vertices. A change in the order corresponds to an EEE-event of some type. Since no such event can occur within **I**, the ordering of 1D surfaces cannot change. This ordering can be done during the off-line construction phase, making the on-line computation of the visible portions of the edge for interval **I** efficient.

The length of the list of pointers to 1D curves for any particular interval **I** varies depending on the amount of occlusion. For an unoccluded edge, the list of pointers to bounding curves is length two, i.e., a curve for each endpoint. Due to occlusion, however, a single edge can be fragmented into arbitrarily many disjoint pieces. An example of this type of fragmentation is shown by the interval $\mathbf{I} = [\varepsilon_{i+1}, \varepsilon_{i+2}]$ in Figure 3.7. Notice that the order of the list of 1D curves for interval $[\varepsilon_{i+1}, \varepsilon_{i+2}]$ in Figure 3.7 determines the appearance of $\mathbf{E}_1$ for all viewpoints within $[\varepsilon_{i+1}, \varepsilon_{i+2}]$. Because of the fragmentation of edges at various viewpoints, the resulting length of the list of pointers to curves bounding the visibility of the edge over a particular interval may be of length greater than two.

To analyze the complexity of the construction phase, let **e** represent the number of edges in a scene. The construction algorithm will form a 2D surface corresponding to each edge in the scene. Let **q** be the number of 1D surfaces on a particular 2D surface. The 1D surfaces are the curves that represent the intersection in aspect space of two 2D surfaces. For a particular 2D surface there are $2 + 2\mathbf{q} + \frac{\mathbf{q}^2 - \mathbf{q}}{2}$ events that can arise in the worst case. This follows by counting the number of each type of event that can occur given a 2D surface:

2	Appearance and disappearance of the entire edge
$2\mathbf{q}$	EV-events (**q** 1D curves, each endpoint an EV-event)
$\frac{\mathbf{q}^2-\mathbf{q}}{2}$	EEE-events (**q** 1D surfaces intersecting on a 2D surface)

It follows that the complete construction of the asp for a single edge (a 2D surface) is bounded by O($\mathbf{e}\,\mathbf{q}^2$). In order to construct a complete 2D surface for every edge in the scene, the cost is O($\mathbf{e}^2\mathbf{q}^2$). In the worst case, $\mathbf{q} = \mathbf{e}$ for every 2D surface so that the construction time is O($\mathbf{e}^4$). In practice, $\mathbf{q}$ is only a small fraction of $\mathbf{e}$ for each 2D surface, and hence the overall construction time is somewhere between O($\mathbf{e}^2$) and O($\mathbf{e}^3$). Notice that, given a particular edge e_i, the computation of the edges that occlude e_i is in the worst case O($\mathbf{e}$). This implies that the construction algorithm can do no better that O($\mathbf{e}^2$) unless other information is given about which edges are "in front of" other edges. For example, if the scene is known to be convex, the construction time of the asp is linear in $\mathbf{e}$ [Plan88].

3.2. Event-Based Interactive Display

The asp construction phase computes a representation off-line that can be used to interactively generate the image sequences of a polyhedral scene from the position of a moving viewer with hidden-lines removed [Plan88, Plan90a]. The asp encodes the viewpoints within the space of all possible viewpoints where visual events occur. The interactive display process allows the user to select a path through the space of all viewpoints, and then uses the asp to determine where along that path changes in the appearance of the scene will occur. In the case where the space of viewpoints is restricted to a great circle $\mathbf{S}^1$ on the view sphere $\mathbf{S}^2$, the user can interactively move along the great circle in either direction and at any speed.

3.2.1. Visible Line Display

The visual events that are computed from intersections in aspect space are edge events and determine the appearance of the lines in the scene. The display phase makes use of these precomputed visual events for the scene as well as a list of global visual events and a list of currently visible faces. Global visual events are those viewpoints where an entire face either appears or disappears. This list is used to focus the display process on only those faces (and hence edges) that are visible at a given viewpoint. The list of currently visible faces (CVF list) is maintained in conjunction with the global event list. The CVF list must be computed for the initial viewpoint of the chosen viewpath. Subsequent viewpoints along the viewpath where faces appear and disappear are stored in a global event list of updates to the CVF list.

With each edge in the scene is associated a fully constructed 2D surface in aspect space that is computed during the construction phase of the algorithm. Each edge is associated with its corresponding 2D surface. The 2D surface represents for its

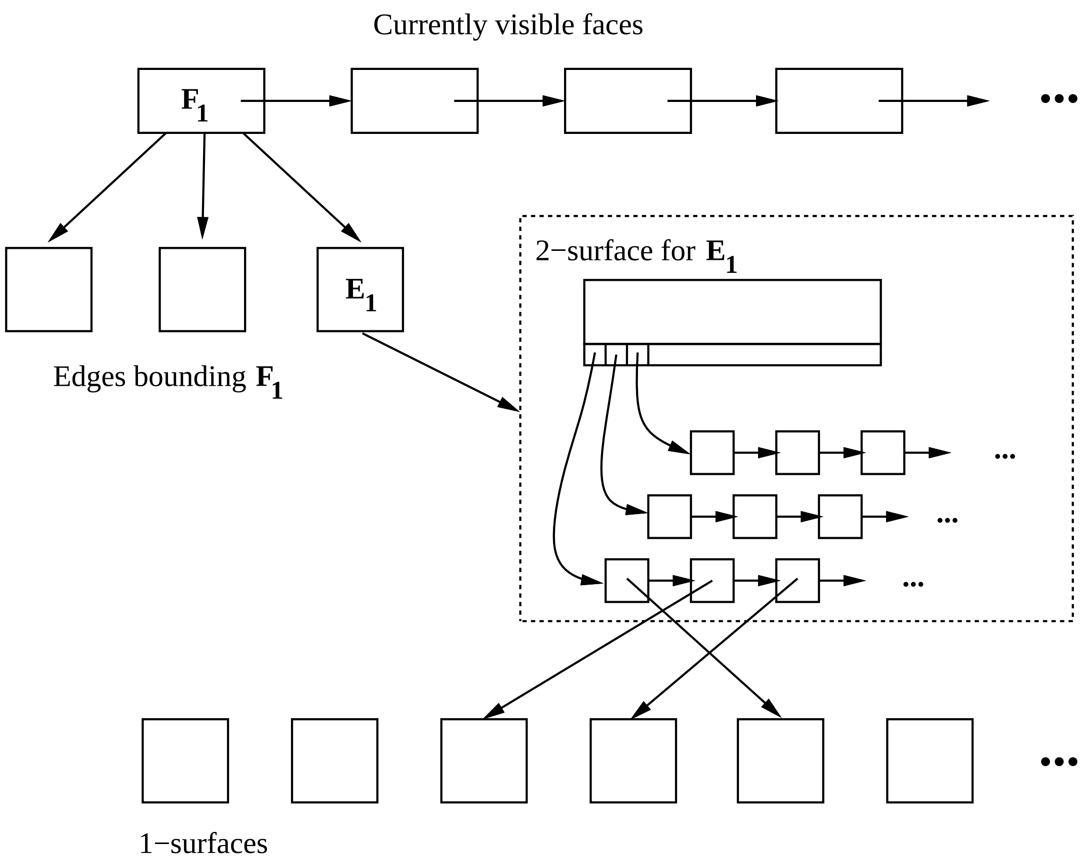


Figure 3.8. The representation used for the display of a polyhedral scene from viewpoints along a 1D path using precomputed visual events.

corresponding edge, the visual events along the viewpath that will affect its visibility. The 1D curves for each 2D surface provide the information needed to display the edge as it appears. The ordering of the 1D curves for each event interval in the 2D surface gives the appearance of the edge directly. Figure 3.8 shows the structure of the representation used to display the scene from the viewpath.

The algorithm for display based on the representation in Figure 3.8 can be divided into two main steps: (1) display the scene for the current viewpoint, and (2) update the ISG based on a change in viewpoint. These steps are iterated, once for each frame in the display. This algorithm assumes the existence of the CVF list for the initial viewpoint and the global event list for the faces in the scene. These two lists can also be precomputed since they depend only upon the initial viewpoint of the viewpath.

The CVF list is a list of faces that are visible from the current viewpoint. Given a viewpoint, backface culling gives the initial list of visible faces. Forward-facing faces are only potentially visible since they might be occluded by other faces. The global event list records the values of θ where all faces in the scene appear and disappear. Faces are oriented and hence viewpoints in front of a face are those from which the face is visible. The viewpoints where the face becomes visible and where it becomes invisible are the two viewpoints where the face appears edge-on. These directions can be computed easily from the directed normal to each face. The global event list is sorted by viewpoint, and with each value is associated the type of event (*appear* or *disappear*) and the face affected.

The display phase begins after the CVF list and the global event list are computed. For each face on the CVF list, its appearance is computed and displayed. The appearance of the edges for each face on the CVF list is computed from the 2D surfaces corresponding to those edges. Each 2D surface has a list of event values that have been sorted. The interval that contains the current viewpoint is found, and the list of pointers to 1D curves in aspect space directly gives the appearance of the edge. A *nil* pointer signifies that the edge is entirely occluded. Hence the only computation involved is the scanning of the event list for a 2D surface and the computation of the image coordinates for each of the 1D curves affecting the appearance of the edge. Since the viewpoint typically changes by small increments along the viewpath, a complete scan of the event list within a 2D surface is not necessary at every frame. The position in the event list for the previous frame is saved. The interval that contains the new value of the viewpoint parameter is usually within a single step or two of the previous frame's interval. For example, Figure 3.7 shows part of a 2D surface for an edge $\mathbf{E}_1$. Suppose that at frame k the value of the viewpoint parameter is θ_k, and that $\varepsilon_i \le \theta_k \le \varepsilon_{i+1}$. A pointer to interval $[\varepsilon_i, \varepsilon_{i+1}]$ is stored with the 2D surface for $\mathbf{E}_1$. If frame $k+1$ corresponds to the viewpoint θ_{k+1}, the event interval for the 2D surface that contains θ_{k+1} must be found. The search, however, can begin at $[\varepsilon_i, \varepsilon_{i+1}]$. For small changes in θ, θ_{k+1} is likely to lie within or close to $[\varepsilon_i, \varepsilon_{i+1}]$.

The cost of the computation of the appearance of each frame along a viewpath is determined by the number of faces visible for that frame. Let **f** be the number of faces on the CVF list for some frame, and assume that each face consists of an average of **e** edges. In order to display a frame, the algorithm must compute the appearance of each edge for each face on the CVF list. Consider the following definitions:

s	Cost of scanning the event list within a 2D surface
a	Cost of computing the image points from a 1D aspect space curve
c	Average number of 1D curves determining an edge's visibility
d = **c s a**	Cost of displaying one edge using the asp

Since the appearance of the scene for a single frame is the appearance of each of the edges on the CVF list, the cost of computing the appearance is **f e d**, i.e., linear in the number of edges to be displayed. In practice, **s** is small after the initial frame, and **a** involves only a small constant number of multiplication and addition instructions (and a single division). The quantity **c** is also, on the average, small and hence the cost of computing the appearance of an single edge is a small constant.

In addition to the cost of computing appearance, a cost is incurred to update the CVF list. Let **u** be the cost of a single update to the CVF list. Let α be the average number of updates for a single frame (i.e., the average number of faces that appear and disappear between two frames). Then the cost to update the CVF list for a single frame is α **u**. Assuming a uniform distribution of events and small increments in viewpoint per frame, α will be small. The update cost is bounded by the cost to update the CVF list. Using an appropriate data structure (e.g., a balanced binary tree) this cost is O(log **f**). Thus the total cost for a single frame is **f e d** + α **u**, which is bounded by the linear cost of computing the appearance of the edges visible in a single frame.

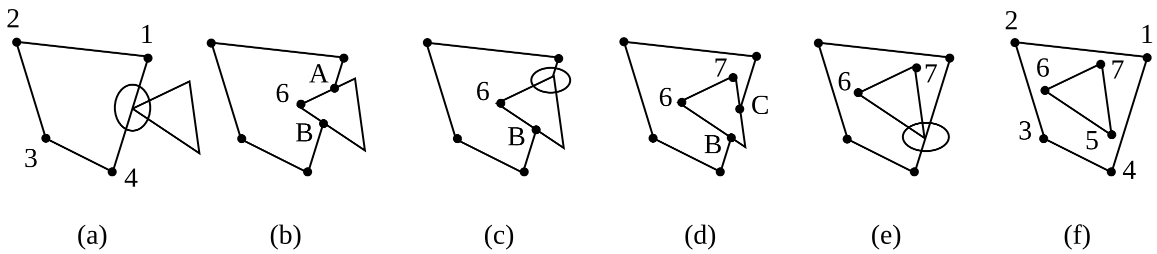


Figure 3.9. The triangle occludes the quadrilateral as the viewpoint changes from left to right. Each circled visual event constitutes an update to the ordered list that preserves the boundary of the visible portion of the face.

3.2.2. Visible Surface Display

The visible-surface problem is a direct extension to the algorithms for visible-line display using precomputed visual events. The significant difference is that for each face the correct edge order must be maintained. This order is maintained for each face independently, so that at each frame the representation of a face is an ordered list of vertices and/or T-junctions. Faces are normally stored as a set of ordered edges, where the ordering gives the directed normal to the face. To maintain a visible-surface representation, this edge ordering around the face is preserved as events occur along the viewpath.

Associated with each visual event is an update to the ordered list specifying where an insertion or deletion should occur. In the visible-line case, it is not necessary to preserve this ordering information since the only concern is the appearance of individual edge segments. For visible surfaces, when an event occurs, the appropriate ordered update that has been precomputed can be applied directly to the list for the affected face. The result is a closed, exact visible surface representation maintained at all viewpoints for each face in the scene.

To illustrate the necessary ordering information, consider the example in Figure 3.9. Each snapshot from left to right shows a different projection of the two faces as viewpoint changes along the viewpath. The circled EV-events in snapshots (a), (c) and

(e) are updates to the ordered list of vertices and EE-events (T-junctions) bounding the visible portion of the face. Immediately before the EV-event in Figure 3.9(a), the list bounding the visible portion of the quadrilateral is { 1, 2, 3, 4 }. The numbers refer to vertices, and the letters (A, B, C) refer to EE-events. The event causes the insertion of the vertex 6 and the EE-events labeled as A and B. The new ordered list after the event in Figure 3.9(a) becomes { 1, 2, 3, 4, B, 6, A }. Similarly, the EV-event shown in Figure 3.9(c) is a deletion of { A } followed by an insertion of { 7, C } at the end of the list. The new list for the visible part of the quadrilateral shown in Figure 3.9(d) is { 1, 2, 3, 4, B, 6, 7, C }. The last EV-event in this example essentially creates a hole in the quadrilateral, so that its visible parts are bounded by { 1, 2, 3, 4 } with a hole bounded by { 5, 6, 7 }.

The extra cost in time and space for maintaining visible surfaces rather than only visible lines is the constant cost resulting from storing and updating the ordered lists for faces. Because insertion and deletion positions can be precomputed, the cost is a small, fixed constant in time and space for each visual event. The benefit of maintaining surface information is to make use of various surface-shading algorithms. The following subsections discuss the issues of shading and shadows.

3.2.3. Shaded Display

The shaded display of polyhedral scenes conveys information about significant features including lighting, shading, surface markings and surface reflectance, texture, and so on. The information about visual events that affect polyhedral surface appearance is already encoded in the asp because the appearance of a face is determined by the appearance of the set of *edges* bounding the face. The description of the appearance of faces that can be extracted from the asp can be used to formulate a hidden-surface version of the previous algorithm for interactive display. The primary advantage of a hidden-surface display is the added realism gained from shading models, multiple light sources and shadows.

The asp is a face-based representation and encodes information about how edges are *connected*. Consequently, the asp represents the appearance of *faces* as a function of viewpoint. By representing the appearance of faces, the area enclosed by a particular face in the image plane is a closed (but not necessarily connected) region corresponding to the projection of a face in $\mathbb{R}^3$. The interior of a closed region can be filled according to shading information based on the reflectance of the surface, the position of the light source(s) with respect to the viewpoint, and the effect of shadows cast by the light source(s) in the scene.

With arbitrary occlusion, the appearance of a single face may be a disjoint set of polygons. This situation is represented in the asp, however, because the asp allows the appearance of a face to be computed as a set of closed polygons, each of which is bounded by an ordered list of edges. Thus the appearance of a face is a set of closed boundaries, even when a face is partially occluded. This set of closed boundaries in the image plane is guaranteed to be "empty" because the asp gives the appearance of the scene with hidden lines removed.

The positions of light sources must be incorporated into the shaded display process. It is significant to note, however, that the asp for a scene as it has been defined is independent of point light source positions. The asp encodes visual events arising from the geometry of the faces and edges in the polyhedral scene. Thus the asp is not dependent on the position in viewpoint space of light sources since the asp represents appearance from all viewpoints. This is significant since the light source can be moved and the scene drawn again without reconstructing the asp. The cost of constructing the asp is incurred only once in an off-line preprocessing phase. It is a natural result of the viewer-centered property of the asp that the position of a point light source defines another viewpoint, and visibility information from all viewpoints is included in the asp.

3.2.3.1. Flat Shading

The shaded display of polyhedral models in the simplest case assigns to each pixel belonging to a given face a fixed intensity value. Assuming a perfectly diffusing surface and point light sources at infinity, the intensity value at each point on the same planar surface is the same. This value is a function of the reflectance model which depends upon the normal to the face, the position of the light sources, and the reflectance properties of the surface. When considering perfectly diffusing surfaces, the viewer's position with respect to the light source does not matter since light is scattered uniformly in all directions. Since the viewing position does not affect the shading of a given face, and the angle between the surface normal and the light source is constant, the appropriate intensity value can be computed just once for each face. This reflected intensity I for a face with surface normal N is given by

$$I = I_a + \sum_{i=1}^{N} k_d I_{L_i} (\mathbf{N} \cdot \mathbf{L_i}) \tag{3.7}$$

where I_A is a constant to account for reflected ambient light, N is the number of light sources, k_d is the percentage of incident light reflected, I_{L_i} is the radiance of light source i, and $\mathbf{L}_i$ is the direction of the ith light source [Whit88].

Under the assumptions of diffuse surfaces and point light sources, the shading model described by Equation 3.7 can be used to precompute the shaded intensity of each of the faces in the scene given a fixed set of light source positions. This information is equivalent to the reflectance map [Horn77] for the scene, and can be used in conjunction with the asp to compute the shaded appearance of each face in the scene from any viewpoint.

3.2.3.2. Interpolated Shading

The shading model defined by Equation 3.7 does not simulate the smoothness of curved surfaces which are approximated by polygons. Given a smooth surface and its corresponding polygonal approximation, the Phong shading model [Phon75] interpolates geometric information to estimate a normal at each point on the surface. This normal is used with an augmented shading function to shade each point on the surface individually, yielding a smoother shading within and across polygons. Notice that with the shading function of Equation 3.7, each point on the same face receives the same intensity value. Using Phong shading, the correspondence between the approximated surface and the polygon is used to shade each point on the polygon independently. This gives a smoother, more realistic shading within each polygon. The Phong model is given by the previous model plus a specular term so that the intensity at each point becomes

$$I = I_a + \sum_{i=1}^{N} k_d I_{L_i} (\mathbf{N} \cdot \mathbf{L_i}) + k_s \sum_{i=1}^{N} I_{L_i} R_s \tag{3.8}$$

where R_s is a reflectance model that approximates specularity and is also related to surface smoothness.

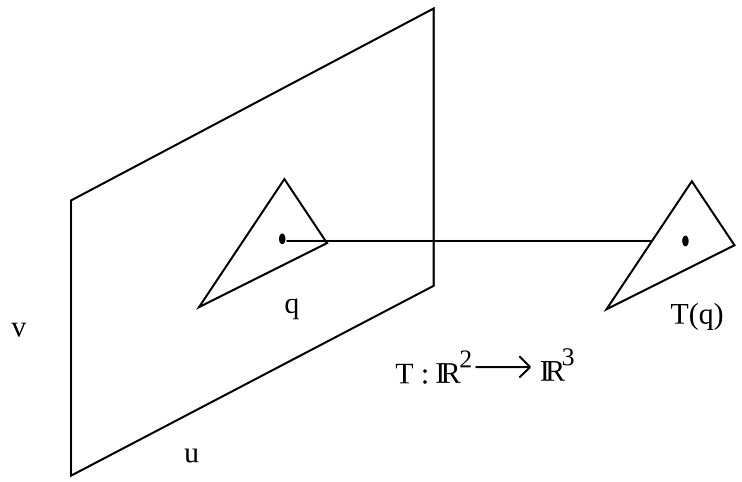


Figure 3.10. The transformation **T** recovers the 3D coordinates on a face in $\mathbb{R}^3$ corresponding to a point **q** within a polygon in the image plane.

The application of the Phong shading model depends on two fundamental geometric assumptions: (1) a smooth, curved surface has been approximated by a set of planar faces, and (2) given a point on a planar face **f** contained in $\mathbb{R}^3$, a corresponding normal to the smooth surface can be estimated. Assumption (1) presents no additional difficulty since the asp deals directly with polygonal representations that may also be approximations of curved surfaces. Assumption (2) can be handled for a point on a face **f** contained in $\mathbb{R}^3$ using a bilinear interpolation scheme that makes use of the normal to the surface at the vertices of **f** [Phon75]. This scheme assumes that the 3D coordinates of the point on the face are known. The information derived from the asp, however, is the appearance in the image plane only, i.e., there is no 3D information. In order to shade a polygon in the image plane using this method, each point to be shaded in the image plane must be associated with a corresponding point on a face in $\mathbb{R}^3$. The transformation **T** that recovers the 3D point from its 2D projection is determined by the equation of the plane in $\mathbb{R}^3$ on which the face lies.

The transformation **T** depends upon the description of the asp for a point in $\mathbb{R}^3$. Specifically, consider a fixed point $\mathbf{p} = (x_0, y_0, z_0) \subset \mathbb{R}^3$. The asp for **p** is computed by considering the image coordinates of **p** as a function of the viewpoint parameters θ and ϕ. The projection onto the image plane can be separated into two parts: a rotation so that the viewing direction is along the z-axis, and the orthographic projection into the image plane. Each component of the rotation is described by an orthogonal 3×3 matrix. Under orthographic projection into the image plane (u, v) we have

$$
\begin{aligned}
u &= x_0 \cos\theta - z_0 \sin\theta \\
v &= x_0 \sin\theta \sin\phi + y_0 \cos\phi + z_0 \cos\theta \sin\phi
\end{aligned}
\tag{3.9}
$$

These equations specify the image coordinates (u, v) of **p** as a function of the two viewpoint parameters θ and ϕ.

The problem of recovering the 3D coordinates of a point given only the values of the two viewpoint parameters and its image coordinates (u_0, v_0) is underconstrained; there are three equations and four unknowns. However, with the equation of the plane in $\mathbb{R}^3$ on which the recovered point is known to lie, the linear transformation **T** is fully determined.

Shading a polygon in the image plane using the Phong shading model involves applying the Phong shading equation (Equation 3.8) at each point within the polygon. By using the transformation **T**, each point in the image plane can be associated with a corresponding point that projects (orthographically) to it from a face in $\mathbb{R}^3$. Figure 3.10 shows the transformation applied to a point **q** in the image plane in order to recover the coordinates of the point $\mathbf{T}(\mathbf{q}) \subset \mathbb{R}^3$. Note that the equation of the plane in $\mathbb{R}^3$ that contains the triangular face must be known in order to apply the transformation. Bilinear interpolation can then be used to approximate the normal to the curved surface at that point. This approximated normal is the normal used in Equation 3.8 to compute the appropriate intensity value for the point **q** in the image plane. Although the Phong model is more expensive than the flat shading model, there are fast approximations to the Phong shading model [Bish86, Duff79].

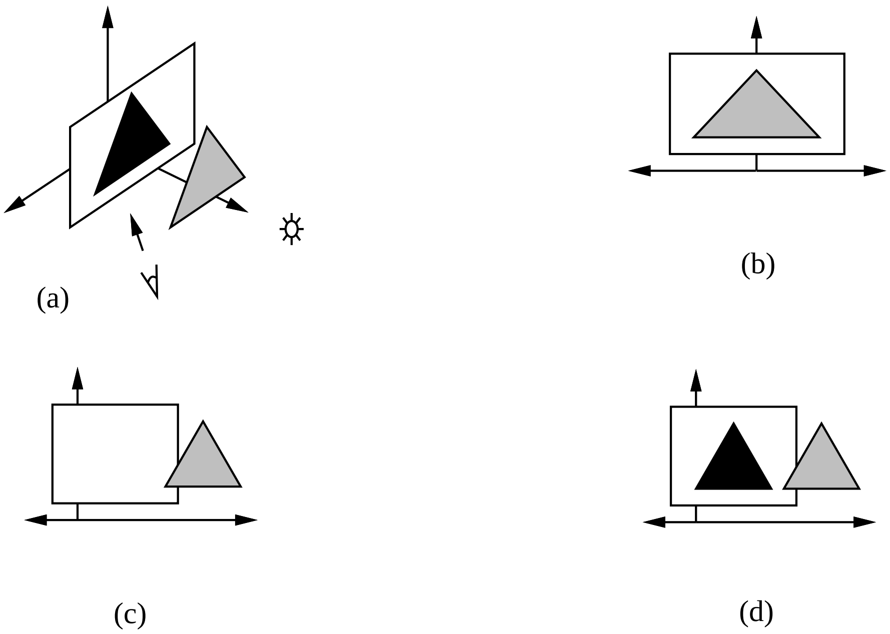


Figure 3.11. A 3D scene containing a single point light source. (a) The 3D scene. (b) The scene projected from the light source direction. (c) The scene as observed from the viewer's position without the shadow. (d) The scene observed by the viewer with the shadow.

3.2.4. Shadows

When the position of the viewer is different from the position of the light source, regions in the scene can lie in shadows. A shadow on a surface results from the occlusion of the light source by some other surface in the scene. Assuming opaque objects, the light source illuminates the first surface it reaches. The surfaces hidden behind those which are illuminated lie in shadow. When the light source position and the viewpoint coincide, there are no shadows. This is true since in that case the set of visible surfaces corresponds identically to the set of illuminated surfaces.

The problem of accurately displaying a shaded scene that contains shadows involves finding the shadow regions in the scene given a viewpoint and the position of each light source. With scenes containing only a single point light source, the problem is that of determining the set of shadow regions. Shadow region computation in polyhedral scenes has been incorporated into hidden surface algorithms using shadow volumes [Crow77] and polygon clipping [Athe78]. Most hidden-surface removal algorithms can compute shadow regions by performing hidden-surface removal from the position of the light source [Joy88]. Rendering can then be performed by integrating the shading of the shadow regions and the visible surfaces according to some shading model. These methods of removing hidden surfaces and finding shadow regions are not designed for interactive viewing. Using these algorithms for displaying a sequence of views of a scene is inefficient since these methods do not make use of viewpath coherence and are not designed for interactive use.

Scenes that contain multiple light sources generate a set of shadow regions for each light source. The interaction between these sets of shadow regions must be computed, i.e., a surface can simultaneously lie in the shadow created by one light source and in the direct illumination of another. The shading of such a surface requires further computation according to the interaction of the sets of shadow regions associated with each of the light sources. Scenes of this complexity have been rendered previously using ray tracing techniques that create shadow regions as a by-product of light-source and ray intersections [Whit80, Cook84].

The asp represents the appearance of each face in the scene from all viewing directions. By treating each light source as an additional viewing direction, the asp can be used to obtain the appearance of a particular face from the light source position. The appearance of a face from the direction of the light source is the part of the face that is under direct illumination. A comparison of the appearances of a face from the viewing direction and from the light source direction gives the location of the shadows on the face that are cast by the light source. As an example consider the illustration in Figure 3.11. The 3D positions of the triangular and rectangular faces are illustrated in Figure 3.11(a). The triangular face casts a shadow on part of the rectangular face. Figure 3.11(b) shows the appearance of the scene from the light source direction. From the position of the viewer, however, the appearance of the scene is different. As shown in Figure 3.11(c), from the direction of the viewer the rectangular face is only partially occluded. Regions visible from the light source are projected from the viewer's position, thus identifying regions of full illumination and shadow (Figure 3.11(d)).

Consider the problem of using the asp to compute the regions of a scene that lie in the shadows created by a single light source. Let **L** and **V** represent the light source position and the viewer position in viewpoint space, respectively. Let **f** be a face in the scene that lies on plane $P \subset \mathbb{R}^3$. Let $\mathbf{f_V}$ be the appearance of **f** from the viewing direction **V**, and let $\mathbf{f_L}$ be the appearance of **f** from the direction of the light source **L**. The appearance of a polygon as encoded in the asp is always a set of closed polygons. Note that because of occlusion, a single polygon may actually appear to be a set of disjoint polygons. Thus $\mathbf{f_L}$ and $\mathbf{f_V}$ represent sets of polygons in the image plane. Furthermore, $\mathbf{f_L}$ represents the part of **f** that is illuminated by the light source **L**, and $\mathbf{f_V}$ represents the part of **f** that is visible to the viewer from the viewpoint **V**.

The computation of regions of the face **f** that lie in shadow hinges on a key observation: the polygons in each of the sets $\mathbf{f_V}$ and $\mathbf{f_L}$ can be thought of as the orthographic projection of disjoint polygons in $\mathbb{R}^3$ that lie on the same plane containing **f**. That is, a transformation can be applied to the edges and vertices in the image plane for both $\mathbf{f_V}$

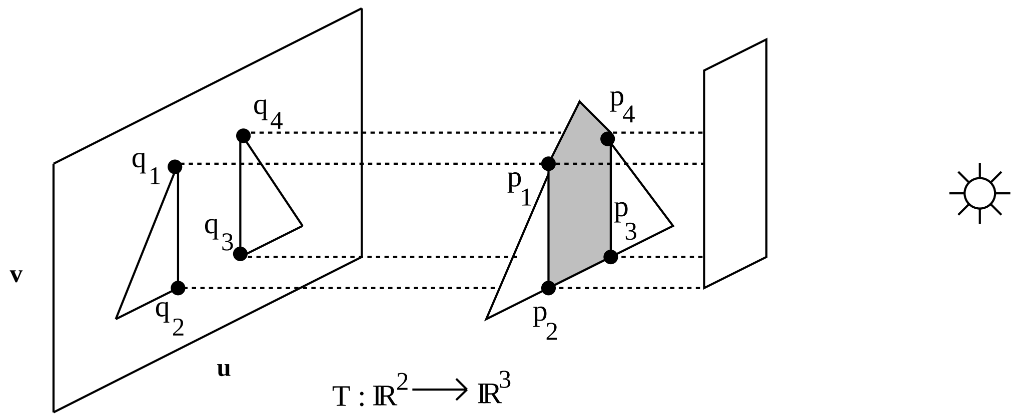


Figure 3.12. The regions of shadow and full illumination on the triangular face can be computed using the appearance of the face in the image and the transformation **T**.

and $\mathbf{f_L}$ in order to recover the corresponding 3D coordinates for the image edges. The 3D polygons that correspond to the transformation applied to both $\mathbf{f_V}$ and $\mathbf{f_L}$ are also sets of closed polygons that lie on plane P in $\mathbb{R}^3$. This is important in order to compare the regions of $\mathbf{f}$ that are visible, to the regions of $\mathbf{f}$ that are fully illuminated. Let $\mathbf{T}(\mathbf{f_V})$ and $\mathbf{T}(\mathbf{f_L})$ represent the sets of polygons in $\mathbb{R}^3$ on plane P corresponding to the necessary transformation applied to $\mathbf{f_V}$ and $\mathbf{f_L}$, respectively. Then the intersection $\mathbf{T}(\mathbf{f_L}) \cap \mathbf{T}(\mathbf{f_V})$ represents the regions in $\mathbb{R}^3$ on $\mathbf{f}$ that are both visible and illuminated. The regions described by $\mathbf{T}(\mathbf{f_V}) - \mathbf{T}(\mathbf{f_L})$ represent those areas on $\mathbf{f}$ that are visible but not illuminated, i.e., shadow regions.

The computation of the intersection and difference of these sets of polygons starts with the transformation $\mathbf{T}$. $\mathbf{T}$ can be applied to every edge bounding the projection of a face in the scene from a particular viewpoint. This yields a set of polygons in $\mathbb{R}^3$ on the face. $\mathbf{T}(A_{\mathbf{f}_L})$ is the set of polygons on $\mathbf{f} \subset \mathbb{R}^3$. Figure 3.12 shows the appearance of a triangular face from the direction of the light source in the image plane. The points $\mathbf{q}_1$, $\mathbf{q}_2$, $\mathbf{q}_3$ and $\mathbf{q}_4$ are vertices in the image plane that are a result of the occlusion of the triangular face by the rectangular face. By applying $\mathbf{T}$ to the vertices $\mathbf{q}_i$ in the image plane, the points $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ and $\mathbf{p}_4$ are recovered on the triangular face in $\mathbb{R}^3$. Notice that the edge between $\mathbf{p}_1$ and $\mathbf{p}_2$ as well as the edge connecting $\mathbf{p}_3$ and $\mathbf{p}_4$ are the boundaries of the region on the triangular face that separate the regions of shadow and full illumination.

In order to compute the intersection $\mathbf{T}(\mathbf{f_L}) \cap \mathbf{T}(\mathbf{f_V})$, the 2D appearance of the face $\mathbf{f}$ from the light source viewpoint can be recovered as a polygon in $\mathbb{R}^3$ that lies on the plane containing $\mathbf{f}$ in $\mathbb{R}^3$. This polygon can then be projected (orthographically) from the viewpoint of the viewer. The two sets of polygons can be compared directly in the image plane corresponding to the viewpoint of the viewer. Recall that for a single point $\mathbf{q} \subset \mathbb{R}^2$ the transformation $\mathbf{T}(\mathbf{q}) \subset \mathbb{R}^3$ is the image of $\mathbf{q}$ constrained to lie on plane P. Then the point $\mathbf{T}(\mathbf{q})$ can be reprojected so that it can be directly compared to the appearance of the face $\mathbf{f}$ from the viewing direction $\mathbf{V}$. Transforming the appearance of $\mathbf{f}$ from the direction of the light source in this way corresponds to the appearance of the fully-illuminated regions of $\mathbf{f}$ from viewpoint $\mathbf{V}$.

For scenes with a single light source, the regions of full illumination and the regions of shadow that may divide the appearance of a face $\mathbf{f}$ can be found by applying the above transformation and then performing the intersection and difference operations

$$\mathbf{S}_1 = \mathbf{T}(\mathbf{f_V}) \cap \mathbf{T}(\mathbf{f_L}) \qquad \mathbf{S}_2 = \mathbf{T}(\mathbf{f_V}) - \mathbf{T}(\mathbf{f_L}) \tag{3.10}$$

The set of polygons $\mathbf{S}_1$ is the region of full illumination; $\mathbf{S}_2$ is the remaining part of the visible portion of **f**, that is the part of **f** in shadow.

3.2.5. Multiple (and Moving) Light Sources

The precomputation of visual events using the asp makes explicit all of the visual events that occur in viewpoint space. The intersection of an interactive viewpath with these precomputed visual events in viewpoint space allows the efficient computation of successive frames in the sequence with hidden surfaces removed. This efficiency is a result of both the viewpath coherence and the visual event precomputation. Because the change in the visibility of the surfaces in the scene is represented, the dominant cost of interactive viewing with hidden surfaces removed becomes the cost of applying the shading model along with the cost of the scan conversion of sets of visible surface and shadow polygons.

The on-line algorithm for interactive viewing computes the appearance of the scene from the starting viewpoint using a standard hidden-surface removal algorithm. As the viewpoint parameter changes, visual events along the viewpath occur. With each visual event is stored a description of how the appearance of visible faces in the scene change from the previous appearance. Hence, for every event an update to the current appearance of the visible faces is made. The second frame in the sequence is computed from the first by determining the visual events that occur under the small change in viewpoint, and then modifying the appearance of each face given the events. The viewpath coherence that results from smoothly changing viewpoints guarantees that only a small number of visual events will need to be processed between any two frames of the sequence.

These operations can be accomplished using two data structures: a list of visual events and an image structure graph (ISG). The image structure graph is a graph of faces, edges and vertices that describes the appearance of the scene for a particular viewpoint. The ISG is constructed initially for the first frame of the viewpath, and is modified for each visual event thereafter. The visual event list is generated from the asp and contains a complete, sorted list of visual events and corresponding updates to the ISG. The construction of both the visual event list and the initial ISG is done off-line.

The interactive viewing of a shaded polyhedral scene is a direct application of the precomputed visual event list and the ISG. Consider a scene with fixed light sources and flat-shading. Intensities for each face are precomputed and stored in a lookup table, and shadow regions for each of the light sources can be similarly precomputed. For a single frame of the sequence the task of displaying the scene with shadows and with hidden surfaces removed is a three-step procedure. First, the appearance of each face in the scene from the current viewpoint is computed using the visual event list and the ISG. Second, each shadow set is transformed using the linear transformation **T** according to the current viewpoint parameters. Finally, the appearance of the scene is rendered by combining the shadow sets and the flat-shading value for each face.

Interactive viewing that includes moving light sources amounts to the additional precomputation of the visual event list and ISG along each light source path. The changes in the polygons in the shadow set for a light source can then be computed efficiently by updating the ISG according to its visual event list. For a static viewer and moving light source, the algorithm is the same as above. For both moving viewer and moving light source, two distinct ISGs and visual event lists must be maintained.

Interactive viewing using an interpolated shading model requires that a distinct intensity value be computed for each point inside a given face. Interpolated shading and fast Phong shading can be done relatively quickly [Swan86, Bish86], and a scan line algorithm can again be used to combine information from shadow sets and shaded visible surfaces in the image plane.

3.3. Computational Results

A prototype of both the preprocessing and the on-line portions of the hidden line algorithm has been implemented. The prototype program is written in C and was tested on a DECstation 3100 running Ultrix V2.1. The input to the algorithm is a polyhedral model of a 3D scene represented as a graph structure. A model is a collection of faces with each face is bounded by a set of edges and each edge is bounded by two vertices. The set of eleven test models used in the timing experiments is shown in Figure 3.13. These models were chosen to represent two basic categories of objects: those that approach the worst-case behavior of the asp construction algorithm, and those representing simple 3D objects. The chain links, the layers of squares, the large and small grids, and the spring

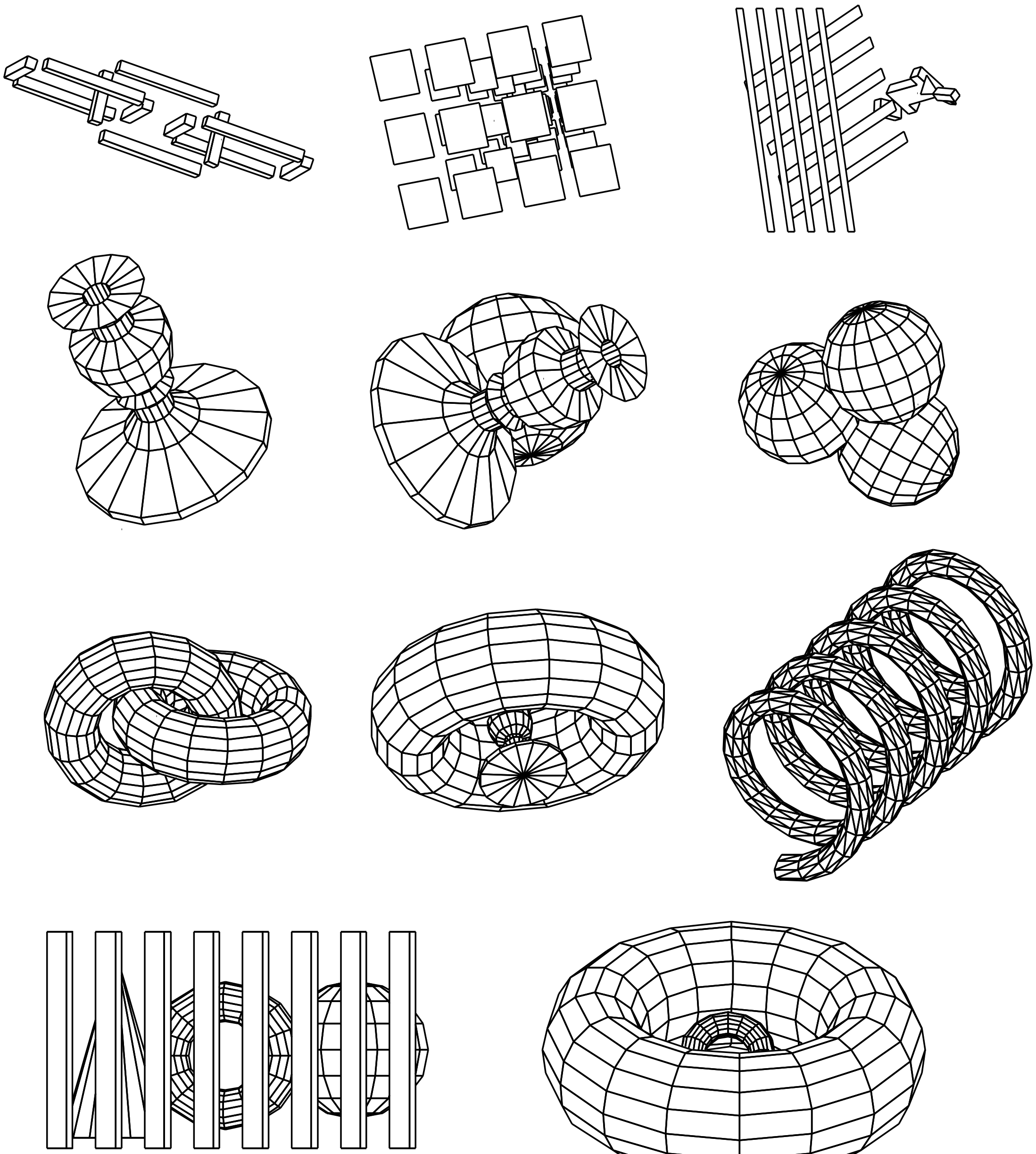


Figure 3.13. The test polyhedra for which the construction and display results are reported.

all contain a great deal of nonconvexity and are visually complex. The models of the tori, spheres and candlestick illustrate the intra-object occlusion created by nonconvexity as well as inter-object occlusion from multiple objects in a scene.

3.3.1. Model Construction

Timing values and sizes for the construction phase of the algorithm for the test models are shown in Table 3.1. The numbers of faces, edges and vertices in each of the models are recorded in the first three columns of the table. For each of the models, Table 3.1 gives the construction time (in seconds) and the size of the resulting representation (in Kbytes). The visual events computed for the edges of the models include three types: the appearance and disappearance of an edge, EE-events (two edges forming a T-junction), and EEE-events (three edges). The maximum, average and total number of events are

Model	Faces	Edges	Vertices	Size (Kbytes)	Time (Seconds)
Chain link	72	144	96	24	10
Layers of squares	48	192	192	28	7
Small grid	41	105	82	36	7
Candlestick	288	560	274	72	130
Candlestick and sphere	416	800	388	120	290
Three spheres	384	720	342	88	143
Interlocked tori	512	1024	512	104	426
Torus and candlestick	544	1072	530	120	484
Spring	1800	2709	909	868	3486
Large grid	449	880	451	216	943
Torus inside torus	512	1024	512	104	436

Table 3.1. Construction times (seconds) and sizes (Kbytes) for the test models.

shown in Table 3.2 and are the sum of these three types of events. The minimum number of events that a model with **n** edges can have is 2**n**, i.e., two events for the appearance and disappearance of each edge.

Several conclusions can be drawn from the data in Tables 3.1 and 3.2. First, the construction times and sizes indicate that the construction of the asp for models with many more faces is indeed tractable. Although the off-line computation time for the largest model, the spring, was approximately 50 minutes, this absolute time is highly dependent on both the prototype program coding efficiency and the hardware configuration. We have estimated that an order of magnitude speedup in the construction time can be achieved by efficient coding to eliminate redundant computation. Second, the number of visual events occurring in a typical scene is only a small constant times the number of edges in the model. Note that the average number of events per edge, even for

Model	Total Events	Events per Edge	Maximum Events per Edge	Seconds per Event
Chain link	996	6.9	21	0.010
Layers of squares	2365	12.3	25	0.003
Small grid	1674	15.9	60	0.004
Candlestick	3109	5.5	25	0.042
Candlestick and sphere	5270	6.6	30	0.055
Three spheres	3733	5.2	22	0.038
Interlocked tori	5135	5.0	18	0.083
Torus and candlestick	5030	4.7	29	0.096
Spring	39954	14.7	52	0.087
Large grid	9776	11.1	122	0.096
Torus inside torus	4395	4.3	27	0.099

Table 3.2. The number of visual events computed for each of the test models. Visual events include the appearing and disappearing of an edge, edge-edge (EE) events and edge-edge-edge (EEE) events.

the spring model, is generally less than 15. The number of visual events is typically about an order of magnitude larger than the number of edges. Extrapolating from the models tested, 10 MB should be sufficient to store the events for a scene with 20,000 to 180,000 edges. However, since the number of events may be worse than linear in the scene size, the maximum possible scene size that uses 10 MB of storage will depend on the visual complexity of the scene and may be much smaller for complex scenes. Still,

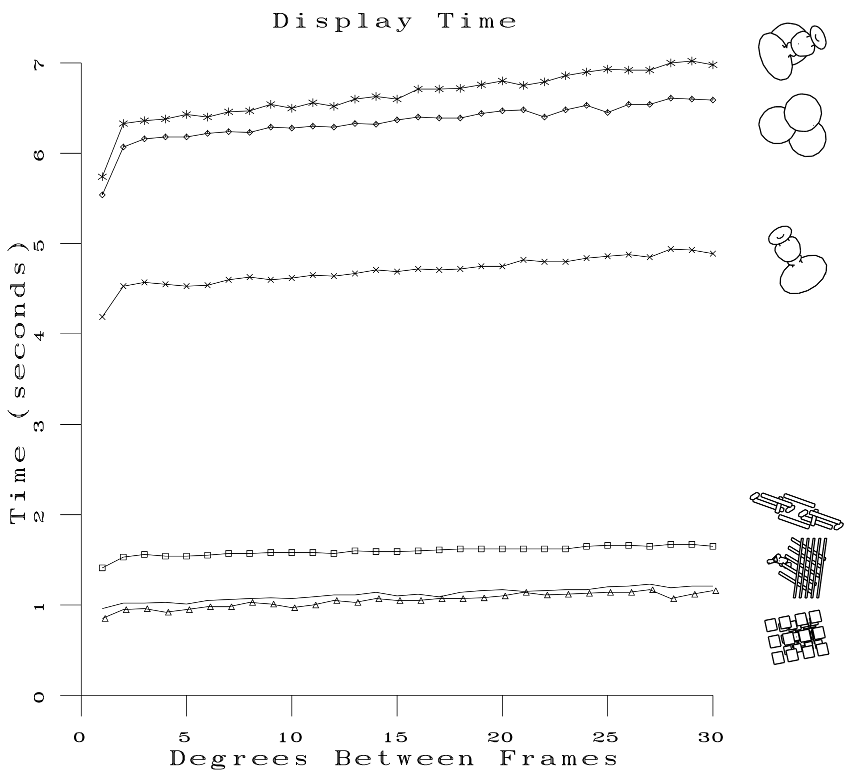


Figure 3.14. The display times for 360 frames of a sequence as a function of the degrees between frames for the first six models from Figure 3.13. The horizontal axis shows the degrees between frames.

the storage requirements appear to be practical for scenes of moderate size and visual complexity.

3.3.2. Display

Timing measurements for the on-line interactive viewing phase of the algorithm are

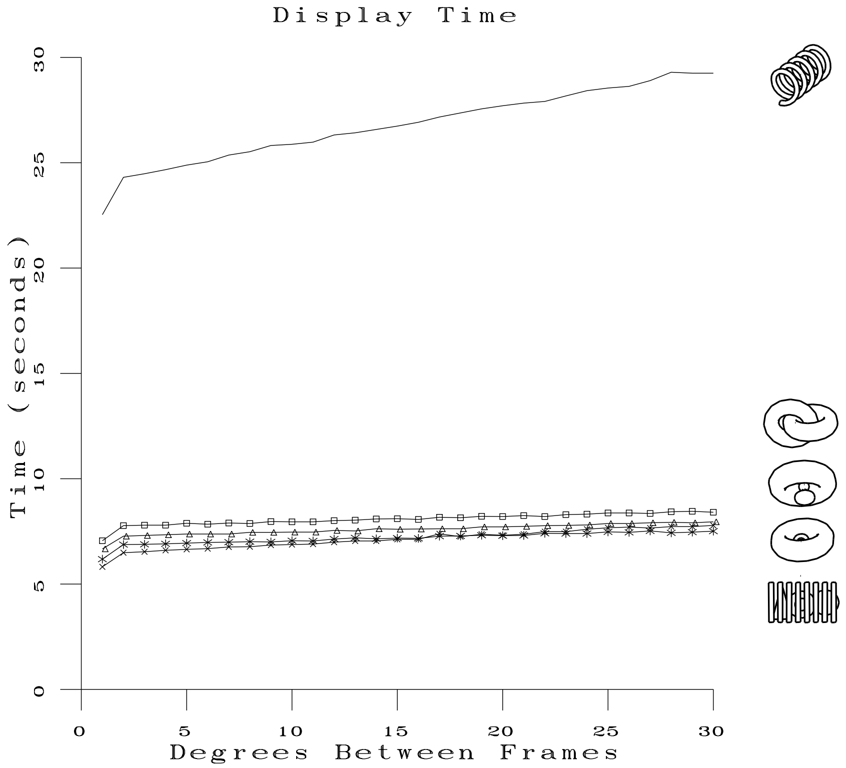


Figure 3.15. The display times for 360 frames of a sequence as a function of the degrees between frames for models 7-11. The horizontal axis shows the degrees between frames.

shown in Figures 3.14 and 3.15. In order to measure the display rate of each of the test models, we measured the time needed to display a sequence of 360 frames. The timing results include the time to update the event structure as well as the time spent on computing the image coordinates of each of the edge segments to be displayed. The viewpath was a great circle (rotation) on the view sphere at $\phi = 0$. Figures 3.14 and 3.15 show the display rates in seconds of the eleven test models. A complete asp and animation sequence was computed and timed for each of these models. Figures 3.15 presents the

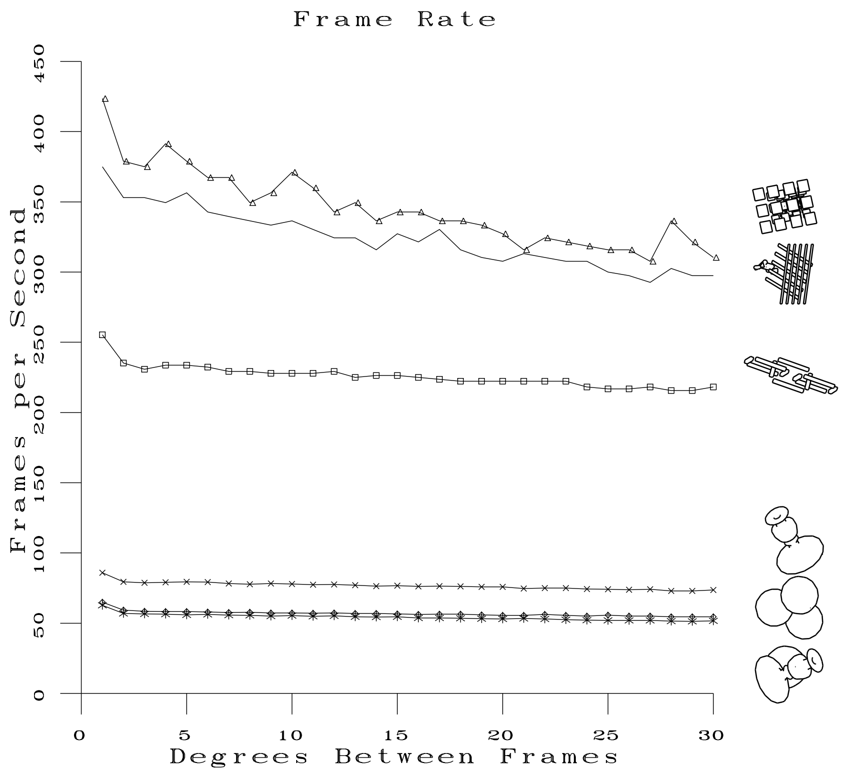


Figure 3.16. Frame rate as a function of the degrees between frames for the display of the test models 1-6.

same timing data in the form of frames per second as a function of the number of degrees between successive frames.

These timing results lead to several observations. First, a small change in the number of degrees between frames in the sequence does not greatly affect the display rate. This indicates that the time needed to compute the image coordinates of the segments to be displayed greatly dominates the time needed to update the event structure. Thus processing a larger number of events between frames has little effect on the frame

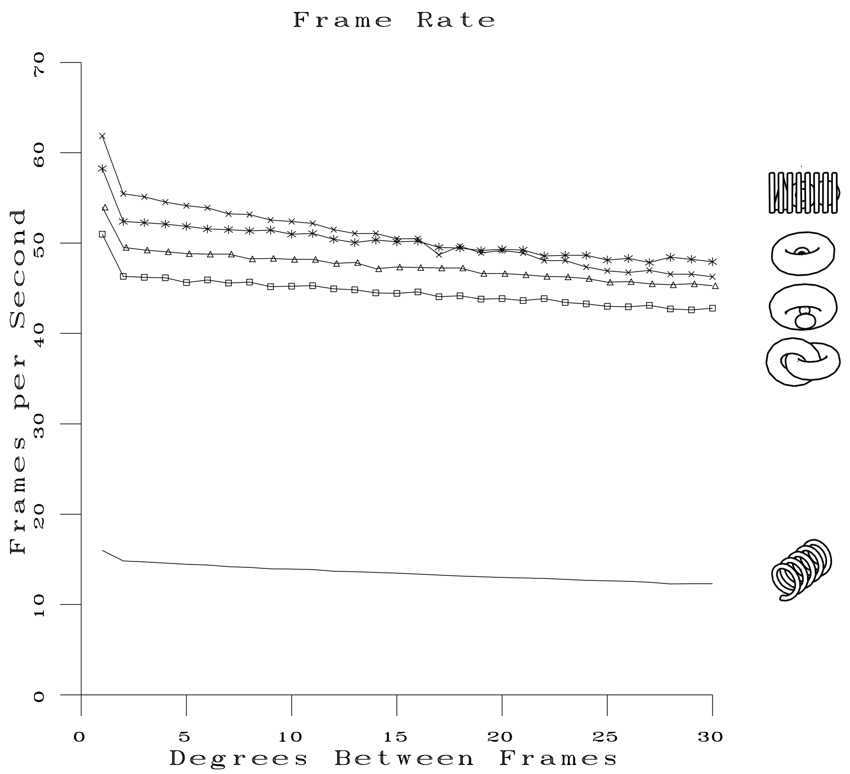


Figure 3.17. Frame rate as a function of the degrees between frames for the display of the test models 7-11.

rate.

The above observations also indicate that the dominant part of the display process is the computation of the image coordinates of the endpoints of segments, and not the processing of visual events. With fast 3D rotation and vector-drawing hardware, this segment coordinate computation can be done quickly for large models. Since the added computational cost of processing visual events is so small, the interactive viewing of much larger scenes can be achieved using the asp.

In summary, our results indicate that the asp can be used to achieve a display rate that is fast enough to allow interactive movement of the viewpoint with scenes containing at least 2,000-10,000 faces on a general-purpose workstation. From the prototype implementation of this algorithm we have found that for scenes of moderate size and visual complexity:

- The number of visual events is in practice a relatively small constant times the number of edges in the scene
- Frame display time changes very little with larger degree increments between frames
- Visual event processing requires less than 5 percent of the total display time

These results suggest that the preprocessing times for larger models will be much lower than the theoretical worst-case bounds and that the resulting size of the event structure will be a relatively small constant times the number of edges in the model. In addition, the small computational cost associated with visual event processing shows that the interactive, on-line display of large models can be achieved when preprocessing time is available.

The interactive viewing of shaded scenes with multiple light sources and shadows is a direct extension of this implementation. The key feature of this approach is the separation of the hidden surface computation in the animation sequence from the application of the shading model and the scan conversion process. The set of object resolution polygons representing the visible faces and the shadow polygons is computed efficiently using the visual event list derived from the asp. Viewpath coherence guarantees that only a small percentage of the visual events need to be processed between a single pair of frames. The difference between hidden-line and hidden-surface computation in this approach is the edge connectivity information that must be preserved, as well as the cost of raster display. The primary cost of shaded display is the cost associated with scan converting the visible surfaces and shadow polygons.

3.4. Hybrid Methods

The methods for visual-event-based interactive display presented here have been studied as a pure method in order to evaluate the tradeoffs and efficiencies that can be obtained. There are several known methods for 3D modeling and for display, however, that can be incorporated into the visual-event framework in order to address some of the weaknesses of this pure approach. The following subsections discuss hybrid methods that address several of the limitations of the pure visual-event-based approach. The problem of the worst-case visual event complexity is solved in part with an algorithm that uses the depth ordering of faces to display visible surfaces. The only visual events necessary to compute and store are those that affect the depth ordering of faces. Thus there is an asymptotic improvement in the complexity of the algorithm when using visual events to compute depth ordering.

3.4.1. Event-Based Depth Ordering

A list of polygons can be displayed with hidden portions removed if the polygons are ordered in depth with respect to the viewer [Newe72]. That is, the surfaces can be scan converted in back-to-front order to cover portions of faces that are hidden [Joy88]. The BSP-tree [Fuch83] is based on this premise: the precomputed tree is a more efficient way to sort the faces of a scene in depth. Given a viewpoint, the depth ordering is available from the BSP-tree by an in-order traversal of the tree. Although the BSP-tree is better than a naive sort, the BSP-tree does not exploit the coherence in the depth ordering between adjacent viewpoints. For viewpoints that are close together, the back-to-front order of the faces may not change, or may only change slightly. In contrast to previous work, we have cast the problem of efficiently determining depth ordering as one where visual events in viewpoint space specify the appropriate change in the back-to-front display order of the polygons with respect to the viewer.

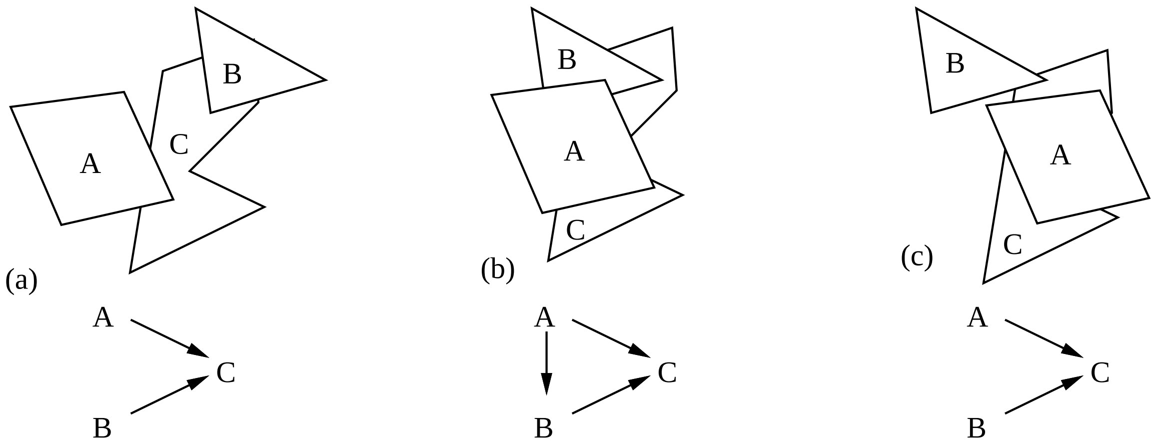


Figure 3.18. The change in the depth ordering between faces A, B, and C is precomputed as a set of updates to the occlusion digraph. A topological sort of the digraph gives the display ordering.

The relationship A occludes B (written as A $\Rightarrow$ B) defines a display ordering between the faces A and B. For a single viewpoint, the occlusion constraints between faces form a digraph. A topological sort of this digraph yields a partial ordering that is the depth ordering necessary to display the polygons. The occlusion graph for the polygons in Figure 3.18(a) yields the digraph shown; face A and face B must both be drawn after face C.

Viewpath coherence is exploited by updating the digraph of occlusion relationships as visual events occur. This digraph maintains the depth constraints from the previous frame and incorporates any that have changed between frames. Obtaining a correct depth ordering for the next frame in a sequence is done by first updating the digraph based on visual events, and then performing the topological sort. The sequence of views in Figure 3.18 shows how constraints are added to the occlusion digraph. In Figure 3.18(b), face A begins to occlude face B so the arc for A occluding B is added to the graph. At this point the topological sort gives the depth order C, B, A because of the added arc between A and B. As viewpoint changes, A no longer occludes B, and so the arc between the two nodes in the digraph is removed (Figure 3.18(c)). This algorithm is very similar in style to the previous visual-event-based algorithms. The key difference is the reliance on the depth ordering to do hidden-surface removal.

Any partial occlusion between two faces must result in a modification of the occlusion digraph. The EE-event (T-junction) is the event corresponding to partial occlusion between two faces. Only EE-events are necessary in order to maintain a correct occlusion digraph, as opposed to the higher-order events necessary for the exact-appearance algorithms presented earlier. This alone represents a substantial savings since there can be as many as O($\mathbf{n}^3$) EEE-events.

There are several subtleties to this algorithm that are straightforward but nontrivial [Kutu91]. One such subtlety involves cycles in the digraph. Cycles that occur in the digraph under certain conditions cause the topological sort to fail (the topological sort can detect the cycle). As shown in Figure 3.19(a), a cycle can arise from a non-convex face with another face, or as in Figure 3.19(b), a set of convex faces can form an

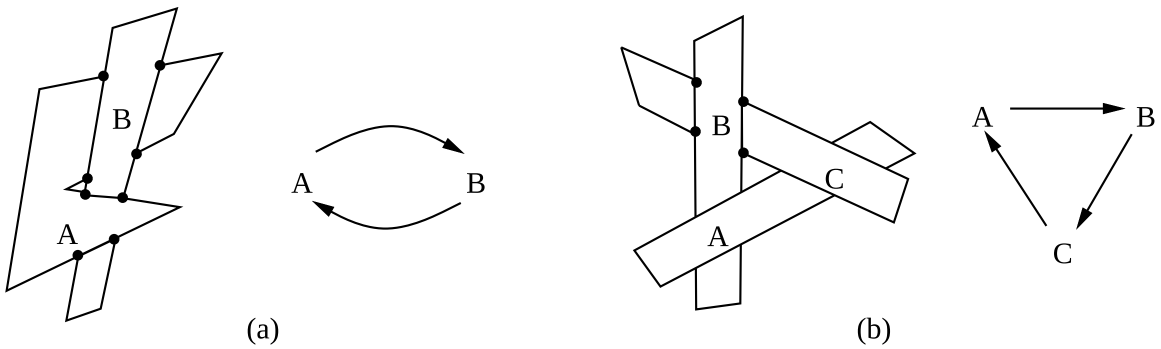


Figure 3.19. The occlusion relationship need not be a strict ordering. (a) Non-convex faces can occlude each other, and (b) cyclic occlusion can occur between 3 or more faces.

occlusion cycle. This is a known problem that all depth-ordering methods must contend with [Fole82]. The BSP-tree avoids the problem by subdividing faces that would potentially form occlusion cycles. Our algorithm detects cycles using the topological sort algorithm. Cycles can then be broken by using EE-events to efficiently compute the intersection in the image plane between two faces in the cycle. The faces in the cycle can be drawn, and the intersection region is drawn last as a correction to the faulty ordering. Figure 3.19(b) shows the vertices where faces B and C overlap in the image plane. These vertices bound the region to be scan-converted after A, B, and C are rendered. The region is drawn as part of B because of the active local constraints between B and C.

This algorithm is a hybrid blend of the depth-ordered method for display and the visual event computation for efficient frame-to-frame coherence. Other depth-ordering approaches do not exploit the coherence available over multiple frames. We are currently implementing and evaluating the performance of the display algorithm for rotations on the view sphere.

3.4.2. Current Work

There is a tradeoff between the cost of precomputation and the cost at display time. For scenes that are not static, the penalty of off-line visual event construction is greater than the gain at display time. The problem of large penalties for recomputing the visual events in a scene that changes only slightly can be solved using a hierarchical structure. The hierarchy divides the scene into components so that the visual events within a component do not affect other parts of the scene. The individual parts of this hierarchy contain visual event information regarding self-occlusion. The components can be combined together using inter-component constraints. We are currently working on the scale and display issues of these coarse-to-fine model hierarchies.

The number of visual events can also be reduced by representing the exact appearance of the occluding contour only rather than every visible edge in the scene. The occluding contour is the projected boundary between the visible portion of a surface and the back-facing portion. This boundary alone carries much of the information for the viewer about the 3D properties of the projected shape. There is a significant reduction in visual events when only considering the occluding contour. An approximate scheme using the occluding contour as a boundary can be used for a faster and more efficient (although approximate) display. Chapter 4 of this thesis develops the rim appearance representation, an exact representation of the visual events affecting the occluding contour. We are working on algorithms for using the occluding contour as a boundary for the inexact but efficient display of large polyhedral models.

3.5. Discussion

This visual-event-based approach can be extended to viewpaths which are not simple great circles. Using the orthographic model, it is possible to compute the asp along any viewpath such that the intersection of that path and the volumes of the asp can be found in closed form. If asp surfaces and their intersections can be represented, the asp can be constructed.

The algorithm can also be extended to perspective projection. In the perspective case it is necessary to be able to find the intersection of the path of viewpoints and the surfaces in aspect space under perspective projection, requiring a straightforward change in the equations listed above. This would be useful, for example, in interactively displaying the appearance of a scene generated by a viewpath moving through a workspace, such as a model of a building [Broo86]. However, depending on the problem, aspect space can become very high dimensional and therefore the asp may require more space and time to compute. For related work on extending the asp to perspective projection, see [Plan86, Plan87, Plan88].

This chapter has presented an efficient algorithm for interactively viewing a polyhedral scene. The algorithm takes advantage of viewpath coherence, a form of frame-to-frame coherence, which is inherent in the sequence of images generated by a moving viewer along a continuous viewpath. The appearance from all viewpoints is computed in a preprocessing phase that works by constructing the asp for the scene. The preprocessing phase also involves computing the initial appearance of the scene with a standard hidden-line or hidden-surface removal algorithm. The on-line phase involves the display of views of the scene with hidden lines or hidden surfaces removed as the user interactively specifies movement in viewpoint space.

The algorithm presented here is practical only for a certain class of scenes. When the number of faces in the scene becomes large and the scene is visually complex, the number of visual events will eventually become too large to store. In the convex case, the number of visual events is O(**n**), where **n** is the number of faces in the scene, but in the worst case the number of visual events is O($\mathbf{n}^3$). However, the prototype implementation shows that for polyhedral scenes of moderate size and visual complexity, the number of visual events is a relatively small constant times the number of edges in the scene. Therefore, in practice, depending on the visual complexity of the scene, a current workstation has enough memory to store the events for polyhedral scenes containing 1,000 to 100,000 edges.

Chapter 4

The Rim Appearance

One of the primary components of 3D model-based computer vision is the representation of the salient, observable features of objects. The occluding contour, which is generated by the projection of points on a surface where the viewing direction is tangent, is one of the primary features of the appearance of an object. The projection mapping generates occluding contours, and the opacity of solid shape causes the creation of T-junctions in the image plane. The arrangement of these contours and T-junctions is directly related to the 3D properties of the shape and provides strong information for recognition [Marr77, Koen84]. Object-centered object models do not explicitly represent the properties of the occluding contour since the occluding contour is not generated by any specific set of object features. Viewer-centered models of 3D shape are better suited to encode the dynamic, viewpoint-dependent nature of the occluding contour.

This chapter presents a novel approach for representing the geometry of the visible occluding contour for polyhedra based on the visual event, defined in Chapter 3. The structure of the occluding contour is modeled in a viewer-centered representation called the *rim appearance representation*. This representation models the occluding contour formed by the edges of a polyhedron that is assumed to be an approximation of a smooth 3D shape as generated under the orthographic projection model. Explicit information is stored about self-occlusion and the appearance in the image plane of the occluding contour. Much of the observable geometry of the self-occlusion of 3D shape is preserved in polyhedral approximations, so it is not necessary to assume a polyhedral world. Further, the geometry of self-occlusion includes viewpoint so that the dynamic changes in occlusion relationships are made more accessible in this representation. The visible occluding contour of a 3D shape is represented as a piecewise-continuous function of viewpoint.

The definition of the occluding contour makes it difficult to represent. The set of points in $\mathbb{R}^3$ on a smooth shape that project to the occluding contour changes smoothly with viewpoint, and hence the changing contour in the image plane is a complicated function of shape, viewpoint and projection. Features such as T-junctions and curvature extrema on the contour are interesting in that they persist over large portions of the space of viewpoints despite the continuous change in the 3D surface points generating them. The evolution of the occluding contour is discontinuous at a finite set of isolated viewpoints where topological changes in the structure of the occluding contour occur. The rim appearance representation encodes an approximation of both the smooth evolution and the points of discontinuity as explicit, connected structures that can be organized and manipulated directly.

Recent work in viewer-centered modeling has concentrated on the aspect graph, a graph enumerating all of the topologically-distinct 2D views of a 3D object, as well as the transitions between views. The aspect graph has been constructed for polyhedra [Plan90, Gigu90, Bowy89], solids of revolution [Egge89, Krie89], and piecewise-smooth objects [Srip89, Ponc90]. The rim appearance representation is different from the aspect graph in two significant ways. First, the rim appearance representation stores only the appearance of the rim, that is, the occluding contour. The behavior of the occluding contour of a polyhedron is a strict subset of all of the topologically-distinct changes in the appearance of the polyhedron. Thus the rim appearance representation is almost always much smaller than the aspect graph for polyhedra. The rim appearance representation models only those features that are observed in the images of smooth objects. A large number of topologically-distinct views in the aspect graph arise from treating the edges of a polyhedron as true surface discontinuities. These artifacts of the polyhedral approximation are omitted in the rim appearance representation. Aspect graphs have been constructed for piecewise-smooth shapes, although the numerical complexity of those representations makes it difficult to extract and represent the features of interest. Our approach avoids difficult numerical problems by relying on the linear features of polyhedra [Faug86].

The second way in which the rim appearance representation differs from the aspect graph is that the rim appearance representation encodes individual features across viewpoint rather than the global topology of the image structure graph. Furthermore, the occluding contour is represented at a level of abstraction above the individual edges forming it. Although the individual edges of a polyhedron and the self-occlusion characteristics of those edges change with viewpoint, the occluding contour itself is represented as an explicit object. The interaction between parts of the occluding contour is represented as a *contour event*, a higher-level event than the interaction of the individual edges of a polyhedron. This abstraction provides a natural way to organize geometric constraints based on the occluding contour. The organization of individual features and feature relationships across viewpoint is lacking in the aspect graph but is included in the rim appearance representation.

This chapter presents the basic properties of the rim appearance representation based on aspect space and the visual event (discussed in Chapter 3). Section 4.1 precisely defines the terms *rim* and *occluding contour*, and presents properties of the rim that make it both desirable but difficult to represent. One primary difference between the rim appearance representation and the complete visual event data computed from the asp is that the rim appearance representation is edge-based rather than face-based. This difference is explained, along with a motivation for the use of the polyhedral model as an approximation of 3D shape for computer vision tasks.

Section 4.2 analyzes the local geometry of the polyhedral rim, and the local and global visual events that create the features of the occluding contour. Section 4.3 presents an algorithm for constructing the rim appearance representation. The details of how to compute features of the occluding contour across a space of viewpoints are presented, and the complexity bounds in time and space for the algorithm are stated. Section 4.4 describes how the exact rim appearance can be approximated. It is shown that the time and space complexity of computing approximate rim information is asymptotically better in the worst case than computing the exact rim information. The tradeoffs of approximating the visual events affecting the polyhedral rim are discussed. Section 4.5 presents the details of an implementation of the construction algorithm of the rim appearance representation, including empirical results from a test set of polyhedral models for which the rim appearance representation has been constructed. Section 4.6 gives concluding remarks with a discussion of the promising future directions of this work.

4.1. Representing Rim Features

The occluding contour produced by a 3D opaque shape is generated by the projection process and is directly dependent on viewpoint. This dependence is a property that gives the contour very strong viewpoint-constraining power. A correspondence between a contour feature and a section of a 3D shape that projects to that contour includes a viewpoint constraint. The dependence upon viewpoint is also the cause of major representational difficulties since, in general, changes in the contour occur even for infinitesimal changes in viewpoint. Section 4.1.1 clarifies the terminology used in defining the rim and the

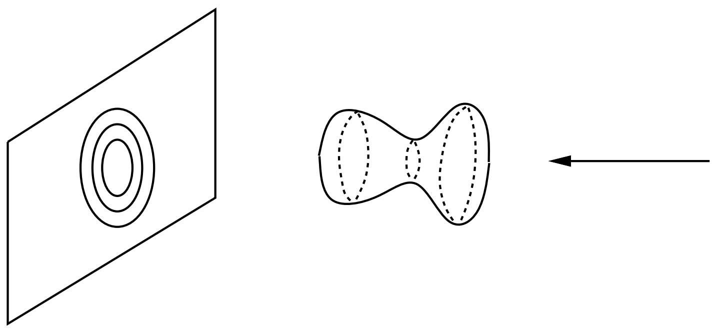


Figure 4.1. The projection of the rim points on a smooth shape (dotted curves) generates a contour in the image (solid curves) that would be generated by a transparent object. The rim points are the boundaries between potentially visible surface points and invisible surface points.

occluding contour. The general rim appearance representation is developed in Section 4.1.2 as a hypersurface in $\mathbb{R}^4$. Section 4.1.3 contrasts the key differences between the rim appearance and the asp, and Section 4.1.4 briefly defends the use of polyhedra as an object model.

4.1.1. The Rim and the Occluding Contour

The terminology for describing the sets of points related to the occluding contour varies widely. Marr used the term contour generator [Marr77], and others have used terms such as limbs [Egge89, Nalw88, Mali87] and the rim [Basr88, Koen87]. To avoid confusion, the precise meaning is defined here for the sets of 3D points called the *rim*, the *visible rim* and the *occluded rim*, and the sets of 2D points in the image plane called the *contour*, the *occluding contour* and the *occluded contour*. Let $p \in \mathbf{S}$ be a point on a smooth, oriented, compact surface in $\mathbb{R}^3$. Let $\mathbf{V}$ be a viewpoint on the unit sphere. A point p is defined to be *visible* when the normal direction at p it is turned toward the viewer, i.e., when $\mathbf{V} \cdot \mathbf{n} > 0$. This is a local definition of visibility for p that does not take into account global occlusion that may obscure p. The rim is a set of points on $\mathbf{S}$, where each point p in this set is defined as follows:

Rim: $\mathbf{V} \cdot \mathbf{n}_p = 0$
p is on the rim if the viewpoint vector $\mathbf{V}$ is tangent to $\mathbf{S}$ at p
Visible Rim: $\mathbf{V} \cdot \mathbf{n}_p = 0;\ \nexists\, q \in \mathbf{S}$ s.t. $q = t\,\mathbf{V} + (1 - t)\,p\ ,\ \ 0 < t < 1$
p is on the visible rim if p is on the rim, and p is visible from $\mathbf{V}$
Occluded Rim: $\mathbf{V} \cdot \mathbf{n}_p = 0;\ \exists\, q \in \mathbf{S}$ s.t. $q = t\,\mathbf{V} + (1 - t)\,p\ ,\ \ 0 < t < 1$
p is on the occluded rim if p is on the rim, and p is not visible from $\mathbf{V}$

The occluding contour is a set of points in the image plane generated from **S** under projection. The contour points are related to the rim as follows:

Contour:	The contour is the projection of the rim
Occluding contour:	The occluding contour is the projection of the visible rim
Occluded contour:	The occluded contour is the projection of the occluded rim

The local definition of visibility divides the entire surface **S** into patches of points that are either potentially visible or not visible. The rim is the transition, or the boundary, between these patches [Koen90]. This definition of the rim is defined by the local visibility condition, and the contour generated by the projection of the rim contains points that are potentially, but not necessarily, visible. The projection of the rim is the contour that would be generated by a transparent shape. Figure 4.1 shows the rim points of a smooth shape and the projection of these points into the image plane.

The projection of opaque shapes causes global occlusion, obscuring some of the rim points. For any viewpoint the set of rim points is only potentially visible, and so the rim can be divided into two sets: the visible rim and the occluded rim. This induces two sets of points in the image under projection. The occluding contour is by definition the projection of the visible rim. The *occluded contour* is the projection of the occluded rim. The two inner closed contours in the image plane of Figure 4.1 are on the occluded contour. The outermost closed loop is the visible part of the contour.

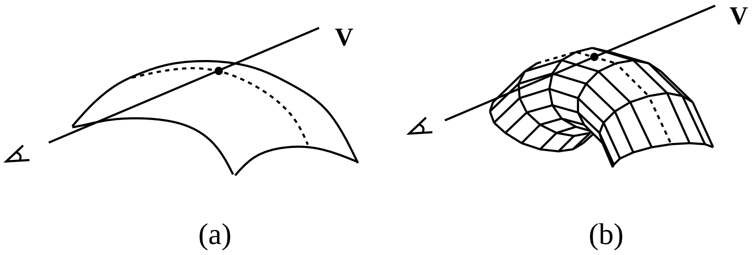


Figure 4.2. (a) The points where the viewpoint **V** is tangent to the surface are rim points. The rim is the boundary between potentially visible surface points and invisible surface points. (b) The polyhedral analog: rim edges are on the boundary between potentially visible and invisible faces.

Given these definitions, the analog of the rim for polyhedra can be defined. The polyhedron is restricted for simplicity to form a set of closed, oriented 3D volumes. Each edge is the intersection of exactly two faces and each face is oriented. An edge is on the rim when exactly one of the two faces adjacent to it is turned toward the viewing direction. This is analogous to the definition of the smooth rim in that the rim edges are the transition edges between parts of the model that are potentially visible and parts that are not visible. In Figure 4.2(b), the rim edges are exactly those edges on the transition boundary between faces that are potentially visible and faces that are not visible.

The rim points for a smooth surface always form a set of closed space curves [Koen90]. Likewise, the rim edges for a polyhedron form a set of closed curves made up of connected "chains" of edges in $\mathbb{R}^3$. The geometry of the rim is relatively simple, and is related to the topology of the surface itself. A torus produces a set of two disjoint rim curves in $\mathbb{R}^3$, one for the outer boundary and one around the hole. Note that the geometry of the rim (the rim is the preimage of the contour under the projection map) is very simple compared to that of the contour. Under projection the closed curves of the rim usually self-intersect.

As a more complicated example, Figure 4.3 shows a cylinder that has been stretched and connected into a trefoil knot [Burd85]. This surface produces two disjoint rim curves

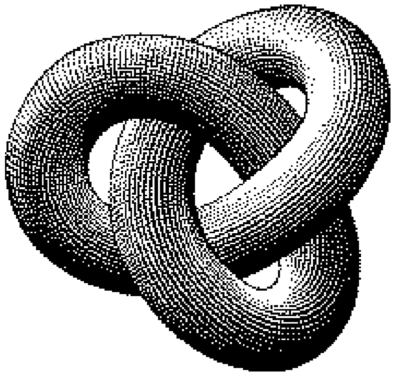


Figure 4.3. The topology of the surface is related to the topology of the rim. In the case of the trefoil, the rim is knotted from all viewpoints.

that are both also knotted. Although the topology of the rim and the surface are clearly related, the depth of the relationship is not currently known.

4.1.2. The Rim Appearance Representation

The rim appearance representation is the organized encoding of all the visual events affecting the appearance of the rim. As presented in Chapter 3, a visual event includes T-junctions and triple-edge junctions, and the degenerate events such as the alignment of an edge and a vertex. The important new contribution of this representation is the fact that the exact appearance of the occluding contour is computed and stored, with its features made explicit.

The rim appearance representation can be formally defined as a surface in aspect space for smooth parametrized surfaces. Consider a surface patch **S** that is parametrized by the mapping $\mathbf{X} : \mathbb{R}^2 \Rightarrow \mathbb{R}^3$ where $\mathbf{X}(u,v) = (\mathrm{x}(u,v),\ \mathrm{y}(u,v),\ \mathrm{z}(u,v))$. The rim points on **S** are dependent on the viewing direction and are expressed by the set of points $\mathbf{R} = \{\mathbf{N}_p \cdot \mathbf{V} = 0\ \}$ where $\mathbf{N}_p$ is the oriented normal at $p \in \mathbf{S}$ and **V** is a vector on the view

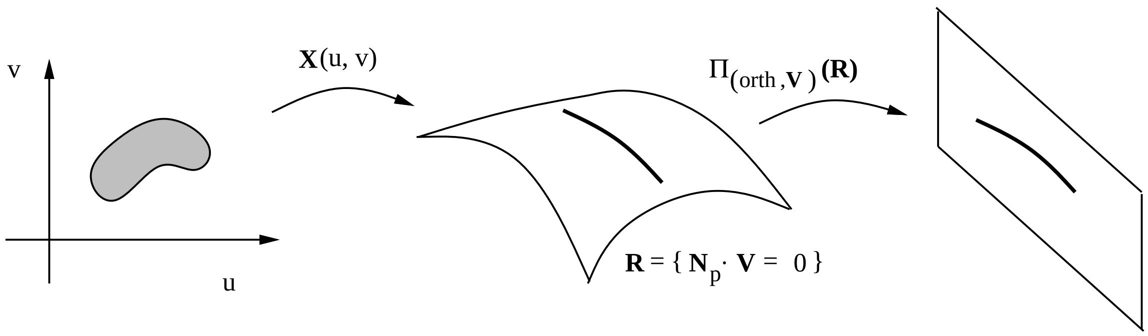


Figure 4.4. The mappings from the parameter space for a smooth surface to the image plane forms a 4-dimensional surface. This surface has singularities where there is occlusion in the image plane.

sphere. The orthographic projection $\Pi_{(orth,\ \mathbf{V})}$ of the set **R** maps the rim points on **S** to the image plane. The projected rim for the complete space of viewpoints for the surface patch **S** forms a hypersurface in $\mathbb{R}^4$ described by $\Pi_{(orth,\ \mathbf{V})}\ \{\mathbf{N}_p \cdot \mathbf{V} = 0\ \}$. Figure 4.4 shows these mappings.

Figure 4.5 gives an intuitive picture of this rim surface in aspect space. The space of viewpoints has been restricted to a great circle on the view sphere. This restriction yields a rim surface in $\mathbb{R}^3$, illustrated on the right in Figure 4.5. The selected viewpoints $\mathbf{V}_{1,}\mathbf{V}_{2,}\mathbf{V}_3$, and $\mathbf{V}_4$ correspond on the left to the contours along the viewpoint axis on the right. The surface that is swept through the aspect space ($\mathbb{R}^3$) can be intuitively thought of as a surface formed by occluding contours that are stacked together at each viewpoint along the path. The interesting thing about this surface is that its singularities correspond to self-occlusion in the image plane.

There has been recent research in formulating an explicit expression for the locus of points that lie on the folds and creases of the multi-dimensional rim surface. Kriegman and Ponce [Krie90] have used parametric patches under a weak perspective viewing model to give an implicit equation for the occluding contour of surfaces of revolution. The implicit equation for a torus under orthographic projection is an 8th degree polynomial containing over 170 terms. Others have computed the singularities of the rim surface for the purposes of computing the aspect graph for surfaces of revolution [Egge89, Krie89] and for parametric surfaces [Ponc90, Srip89]. All of this work with smooth model representations suffers from difficult numerical problems.

Numerical difficulties are avoided when polyhedra are used because of the linearity of edges. The rim surface for polyhedra affords a local way of expressing the singularities as piecewise interactions related to individual edges. Each edge corresponds to a volume in aspect space, the boundaries of which are algebraic surfaces and curves with a geometric interpretation that results from the apparent intersection in the image plane of pairs and triples of unconnected edges. The details of this local, piecewise representation of the rim surface are presented in this chapter. A complete discussion of aspect space itself can be found in the work of Plantinga and Dyer [Plan88, Plan87].

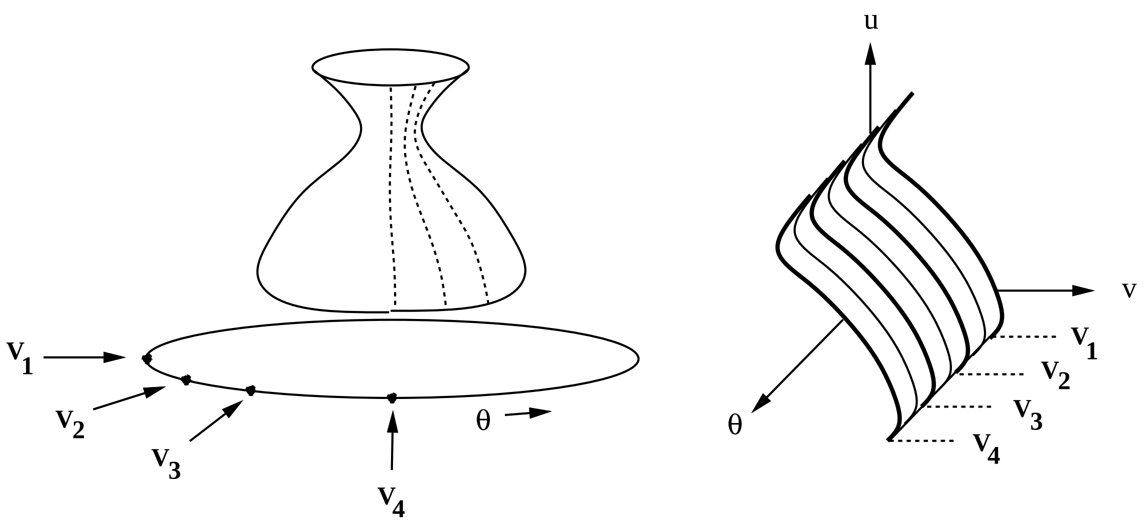


Figure 4.5. Viewpoint is restricted to 1 degree of freedom to produce an occluding contour surface that is 3D.

4.1.3. An Edge-Based Representation

The asp for polyhedra stores visual events affecting every face in the model. As presented in Chapter 3, the complete visual event information can be useful for the exact display of models over a sequence of viewpoints. The construction of the asp has been cast as an intersection problem, where the visual events are computed by considering the asp for individual faces and the way those asps intersect in aspect space. This approach can be thought of as face-based in the sense that the resulting description gives the exact appearance of individual faces.

There is a fundamental difference between the face-based asp computation and the rim appearance representation. The rim for a single viewpoint is a set of edges, most of which do not lie on the same face. The important thing to be computed is how the rim edges are connected to each other as a part of the rim. The appearance of faces is not important, but rather the appearance of edges that are constrained to lie on the rim. Thus the rim appearance representation is edge-based rather than face-based, and this difference is manifested in the way the visual events affecting the rim are computed. The particular details of these differences will become clearer when the construction algorithms are described in Section 4.3.

4.1.4. Shape Approximation

The class of polyhedral 3D shapes is the earliest and most fundamental representation for solid shape in computer vision and computer graphics. Pioneering work in computer vision by L. Roberts, for example, and early work in realistic image synthesis was based almost solely on the geometry of polyhedra. Much of the work in 3D recognition still relies on the simplicity and compactness of the polyhedron (for example, see [Thom87, Murr89]). Experience in both computer vision and graphics has shown that the speed and simplicity of linear representations can compensate for size increases and approximation error, provided the appropriate feature information is retained [Faug86, Lowe89]. In the present context, it is important to preserve the occluding contour and the interactions between contours as a result of self-occlusion. The local approximation of smooth surfaces with planar patches preserves these contour features while affording linearity. Further, the approximation can be made arbitrarily close by selective linear refinement where necessary.

Polyhedra eliminate the continuous property of the rim because the set of rim edges changes discretely as viewpoint changes. These characteristics make polyhedra acceptable as the basis for an explicit, approximate model of the occluding contour.

4.2. The Geometry of Self-Occlusion

The underlying model for the rim appearance representation is the polyhedron. The geometry of a polyhedron is simple because of its linearity and compact representation, and the explicit form of the edge-vertex and EEE-events are at worst quadratic. It is assumed that the polyhedron is an approximation of a smooth object. Thus the numerically complex occluding contour can be approximated as a piecewise function of linear edges interacting with each other in the image plane.

This section is divided into three parts. First, the geometric conditions on the polyhedral rim are precisely defined. A polyhedral rim edge is defined by the local orientation of the surfaces that meet to form the edge. This definition approximates the behavior of the smooth rim as viewpoint changes. Second, the visual events causing

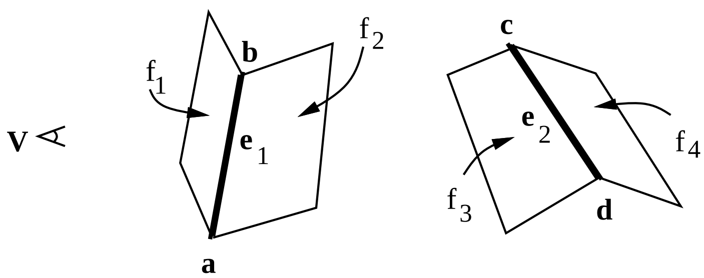


Figure 4.6. Two edges defined by the faces that meet to form them.

topological changes in the projected line drawing of a polyhedron are reviewed. Finally, the geometry of a T-junction formed by the apparent intersection of two edges is presented. These geometric properties are the basis for the algorithm to compute the rim appearance representation described in detail in Section 4.3.

4.2.1. The Polyhedral Rim

The faces of a polyhedron are planar and meet at edges where there is a surface orientation discontinuity. A face does not turn away from the viewer smoothly, but rather disappears instantaneously at the viewpoint where the face is edge-on to the viewer. The smooth occluding contour is approximated by defining an edge to be on the rim when the two faces that meet at the edge are oriented such that exactly one of the faces is turned away from the viewer.

The geometric conditions for a rim edge can be specified in terms of edges, vertices and surface normals. Specifically, let e_1 and e_2 be two edges in $\mathbb{R}^3$ as shown in Figure 4.1. The faces f_1 and f_2 meet to form e_1, and the faces f_3 and f_4 meet at edge e_2. We denote the directed unit normal to f_i as $\mathbf{n}_i$, and a viewpoint on the unit sphere, directed away from the origin, as $\mathbf{V}$. A rim edge is defined as follows:

Definition: An edge of a polyhedron is on the rim if and only if exactly one of the two faces forming the edge is visible, i.e.,

$$e_1 \text{ on rim} \Leftrightarrow (\text{visible } (f_1) \text{ XOR visible}(f_2))$$

f_i is visible from $\mathbf{V}$ means that $\mathbf{V} \cdot \mathbf{N}_i > 0$. For example, e_1 in Figure 4.6 is on the **rim** when either of the following two geometric conditions hold:

$$\begin{array}{l} \mathbf{V} \cdot \mathbf{n}_1 > 0 \\ \mathbf{V} \cdot \mathbf{n}_2 < 0 \end{array} \quad \text{or} \quad \begin{array}{l} \mathbf{V} \cdot \mathbf{n}_1 < 0 \\ \mathbf{V} \cdot \mathbf{n}_2 > 0 \end{array} \tag{4.1}$$

These rim constraints define the rim locally. Since rim edges can interact globally, this definition does not say anything about the visibility of e_1 from $\mathbf{V}$. Hence the local definition of a rim edge must be augmented with the geometric constraints that arise from the global occlusion that can occur under projection.

4.2.2. Visual Events

The projected edges of a polyhedron form a line drawing in the image plane. Viewpoints where the connectivity, or topology, of the line drawing changes are viewpoints where visual events occur. A viewpoint that causes a topological change in the line drawing from any infinitesimal change in viewing direction is called a visual event. It has been shown that all such visual events can be found for polyhedra by computing the edge-edge-edge event (EEE-event) [Plan90, Gigu90]. The EEE-event is the apparent intersection of three not necessarily adjacent edges. A degenerate case of the EEE-event is the edge-vertex event (EV-event), where two of the edges actually meet at a vertex. See Section 3.1.4 for more details on properties of EEE-events.

4.2.3. The Geometry of the T-junction

The primary event affecting the appearance of the visible rim is the formation of a T-junction that results from self-occlusion. The T-junction is not an infinitesimal visual event; T-junctions persist in viewpoint space and are bounded by the visual events introduced above. Recall from Section 3.1.2 the distinction between the stability of various visual events (singularities in the projection map). A quantitative representation of a T-junction that occurs as a result of self-occlusion must include both the bounding visual events where the T-junction appears and disappears, as well the quantitative appearance of the T-junction in the image plane. This section analyzes the geometry of a T-junction between two edges. The geometry is used for rim edges to compute the exact appearance of the occluding contour. The goal of computing the interaction between edges that form T-junctions is to encode the dynamic properties of a T-junction, represent the exact appearance of the rim, and to make explicit the dynamic changes that occur in the occluding contour with respect to viewpoint.

There are specific geometric constraints that determine whether two edges in $\mathbb{R}^3$ can form a T-junction from some set of viewpoints. Two edges that form a T-junction can be labeled *occluding* and *occluded* where the occluding edge is the one closest to the viewer. One basic constraint is that only *rim* edges can be labeled as occluding edges.

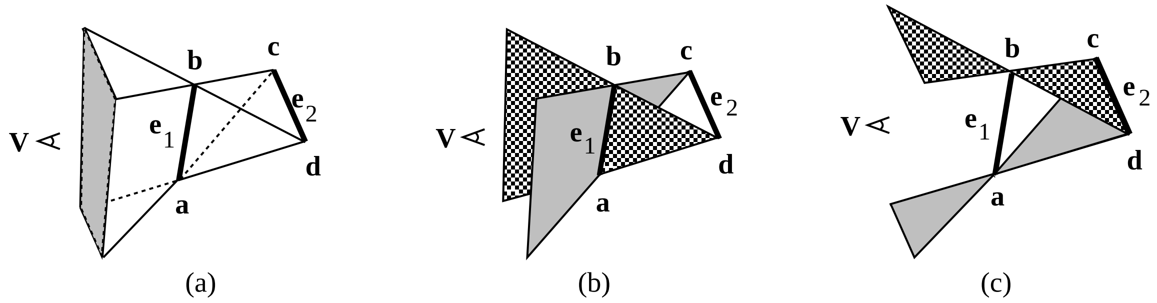


Figure 4.7. (a) The two edges e_1 and e_2 form a T-junction in viewpoint space defined by four planes that form a tetrahedron from the four vertices belonging to the two edges. (b), (c) The boundaries of the space where the T-junction occurs.

Further, only convex edges can be occluding edges. Convex edges are those formed by two faces with an interior angle less than π. This restricts the number of potential T-junctions since only a subset of the edges is convex, and each of these is on the rim from a restricted set of viewing directions.

For smooth objects, the occluded edge can be either a smooth, occluding boundary or a surface orientation discontinuity. Since we are assuming that the polyhedron approximates a smooth object, we restrict the occluded edge to also be on the rim. This restriction can be relaxed to also include edges that correspond to true surface discontinuities in a piecewise-smooth object.

The geometry of the visibility of two edges that form a T-junction in the image plane forms a partition of viewpoint space. Two edges e_1 and e_2 form a partition of space into a region where the edges will project to form a T-junction. The planes that bound this region are defined by the vertices at the endpoints of the two edges. These four planes form a tetrahedron with four triangular sides (see Figure 4.7(a)). Each of the planar sides is defined by a set of three vertices: plane $(\mathbf{a},\mathbf{b},\mathbf{d})$ and plane $(\mathbf{a},\mathbf{b},\mathbf{c})$ share the common edge $\overline{\mathbf{ab}} = e_1$; plane $(\mathbf{c},\mathbf{d},\mathbf{a})$ and plane $(\mathbf{c},\mathbf{d},\mathbf{b})$ share the common edge $\overline{\mathbf{cd}} = e_2$.

A viewpoint that causes two edges to form a T-junction must lie within the region bounded by the planes forming the tetrahedron above. This restriction is expressed by the following four T-junction inequalities:

$$\begin{aligned} [(\mathbf{c}-\mathbf{b}) \times (\mathbf{d}-\mathbf{b})] \cdot \mathbf{V} &< 0 \\ [(\mathbf{c}-\mathbf{a}) \times (\mathbf{d}-\mathbf{a})] \cdot \mathbf{V} &< 0 \\ [(\mathbf{a}-\mathbf{c}) \times (\mathbf{b}-\mathbf{c})] \cdot \mathbf{V} &> 0 \\ [(\mathbf{a}-\mathbf{d}) \times (\mathbf{b}-\mathbf{d})] \cdot \mathbf{V} &> 0 \end{aligned} \tag{4.2}$$

Thus e_1 occludes e_2 from viewpoint $\mathbf{V}$ if and only if (1) both e_1 and e_2 are on the rim (the rim constraints shown in Eq. (4.1) are satisfied), and (2) the viewpoint is inside the region bounded by the four planar sides of the tetrahedron (the T-junction constraints shown in Eq. (4.2) are satisfied). Geometrically, the T-junction inequalities determine the visibility of the four planes defining the tetrahedron. The two faces of the tetrahedron containing e_1 must be visible from $\mathbf{V}$; the other two planes that contain e_2 must not be visible. Visible means that the viewpoint $\mathbf{V}$ is in the plane's positive half-space. The tetrahedron is oriented so that the *outside* is beyond e_1 and e_2, and the *inside* is between them. The viewpoints in the shaded region in Figure 4.7(a) are those that satisfy Eq. (4.2).

Eqs. (4.1) and (4.2) specify the geometric relationship between two edges that potentially form a T-junction. That is, given the set of all $\mathbf{V}$ that satisfies the rim constraints for e_1 and e_2, it can be determined if there is *any* $\mathbf{V}$ that also satisfies the T-junction constraints. A direct and efficient method for solving this problem uses the geometry of the regions in $\mathbb{R}^3$ formed by two edges. The two faces f_1 and f_2 and the two planes defined by $\mathbf{a},\mathbf{b},\mathbf{c}$ and $\mathbf{a},\mathbf{b},\mathbf{d}$ all intersect at e_1 (see Figure 4.8(a)). The viewpoints that satisfy the rim constraints for e_1 are bounded by the planes for f_1 and f_2. The viewpoints that satisfy the T-junction constraints must lie within the two planes defined by $\mathbf{a},\mathbf{b},\mathbf{c}$ and $\mathbf{a},\mathbf{b},\mathbf{d}$. Hence, the intersection of these two regions must be non-empty in order for a T-junction to occur. Call the necessity of this non-empty region condition (1).

Likewise, the two faces f_3 and f_4 and the two planes defined by $\mathbf{c},\mathbf{d},\mathbf{a}$ and $\mathbf{c},\mathbf{d},\mathbf{b}$ all intersect at e_2 (see Figure 4.8(b)). The viewpoints that satisfy the rim constraints for e_2 are bounded by the planes for f_3 and f_4. The viewpoints that satisfy the T-junction constraints must lie within the two planes defined by $\mathbf{c},\mathbf{d},\mathbf{a}$ and $\mathbf{c},\mathbf{d},\mathbf{b}$. Hence, the

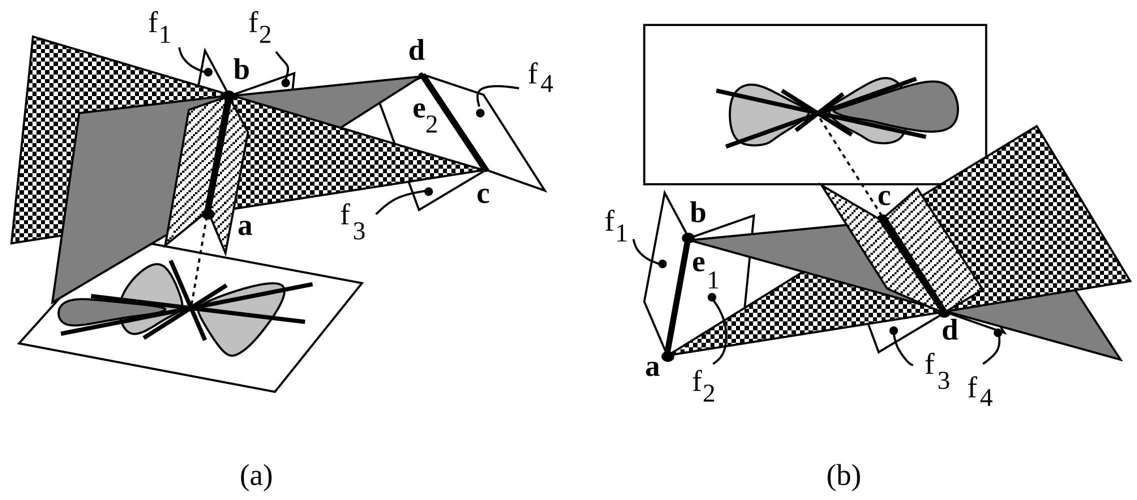


Figure 4.8. (a) The regions on the plane normal to the edge e_1 defined by the intersection of the four planes meeting at e_1 determine whether the rim constraints and the T-junction constraints are satisfied for e_1. (b) The same geometry for the four planes meeting at e_2 determine whether the rim constraints and the T-junction constraints are satisfied for e_2.

intersection of these two regions must also be non-empty in order for a T-junction to occur. Call the necessity of this non-empty region condition (2).

The geometric constraints in Eqs. (4.1) and (4.2) are mutually satisfied if and only if the two edges form a T-junction. That is, all the constraints are satisfied if and only if condition (1) and condition (2) hold. This proves the following:

Theorem: The rim conditions (1) and (2) hold if and only if edges e_1 and e_2 form a T-junction for some viewing direction.

Determining whether conditions (1) and (2) are satisfied for two specific edges is done using the four planes involved with each edge and the plane orthogonal to the edge.

Figure 4.8(a) shows the intersection of the four planes with the horizontal plane normal to e_1. The two large shaded regions on this horizontal plane that are opposite each other represent areas where e_1 is on the rim. The longer shaded area shows where e_1 occludes e_2. Any overlap in these shaded areas in the plane orthogonal to e_1 implies that condition (1) is satisfied. This overlap is computed efficiently by the projection into the orthogonal plane of the vectors in each of the four support planes. The test for this overlap amounts to an ordering of four vectors in the orthogonal plane. Figure 4.8(b) shows the analogous geometry for condition (2).

In summary, the analog of rim points in polyhedra is the set of edges satisfying the geometric property that only one of the two faces meeting at the edge is turned toward the viewer. The visual events that affect a change in the topology of the edges in the line drawing of a polyhedron are completely determined by the EEE-event, where the EV-event is a special case of the EEE-event. Thus the definition of the rim edge can be combined with the visual event computation in order to compute exactly those viewpoints where the polyhedral rim changes topologically. The T-junction is a persistent visual interaction, and the viewpoints where T-junctions are created and annihilated are bounded by the EEE-event. The geometry of the T-junction combined with the definition of the rim provides a fast test to determine whether two rim edges interact to form a T-junction. Once two edges are known to form a T-junction, a quantitative description of the T-junction in the image plane can be computed. This description can be incorporated into an algorithm for constructing the rim appearance representation, which is the subject of the next section.

4.3. Constructing the Rim Appearance Representation

The rim appearance representation models the occluding contour as a function of viewpoint. Representing the rim as a function of viewpoint is related to the asp [Plan88], a complete representation of appearance as a function of viewpoint. The relationship of the rim appearance representation to the asp and the aspect graph depends on a single key component: *aspect space*. Aspect space, the cross-product space of the image plane $\times$ viewpoint space, is the central component in the construction of the rim appearance representation. Aspect space has been shown to be useful for encoding the exact appearance of polyhedra as a function of viewpoint [Plan88], constructing the aspect graph

[Plan90, Plan88], and for the interactive animation of polyhedral scenes [Seal90, Plan90a].

4.3.1. Aspect Space and the Rim Appearance Representation

Aspect space, the cross-product space of the image plane $\times$ viewpoint space, makes use of a multi-dimensional space to explicitly encode the appearance of objects for all viewpoints. The dimensionality of aspect space is dependent on the projection model and the geometry of objects. Aspect space for orthographic projection is 4D since there are two degrees of freedom in viewpoint space and two degrees of freedom in the image plane. A review of aspect space and visual events was given in Chapter 3. A complete discussion of 4D aspect space can be found in Plantinga's thesis [Plan88].

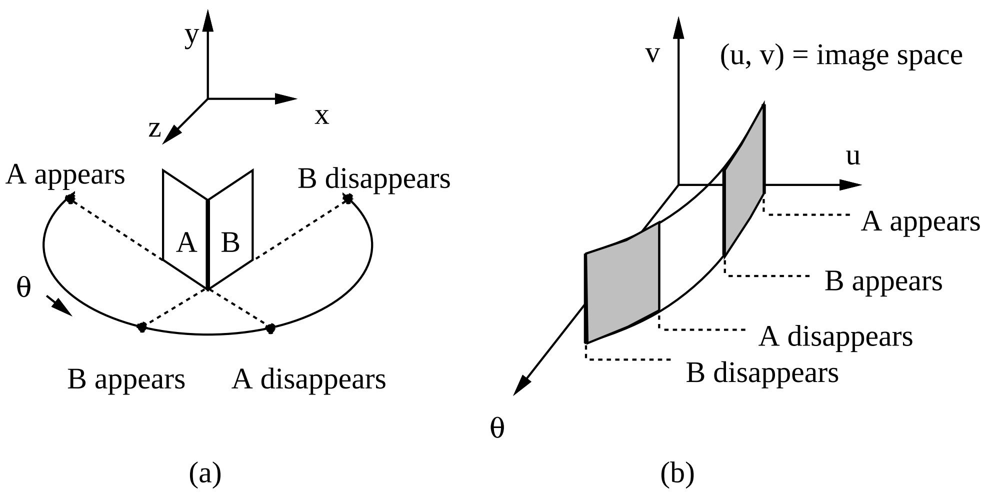


Figure 4.9. (a) An edge formed by two faces A and B. (b) The surface in aspect space corresponding to the edge in (a). The visibility of the edge in (a) as a part of the rim is represented by the shaded parts of the surface.

For a 3D aspect space containing only one degree of freedom in viewpoint, a vertex (x_0, y_0, z_0) generates a 1D curve in aspect space. Points on this curve correspond to the location of the vertex in the image plane for a specific viewpoint. The equations for this curve are derived from the image coordinates of the point after a rotation by θ about the y-axis (see Figure 4.9(a)). Denoting coordinates in the image plane (u, v), the result is

$$u = x_0 \cos\theta - z_0 \sin\theta \qquad (4.3)$$
$$v = y_0$$

These equations specify the appearance of a vertex as a 1D curve in aspect space $(u(\theta), v, \theta)$, $-\pi < \theta \le \pi$.

A line segment connecting two vertices $\mathbf{p}_1 = (x_1, y_1, z_1)$ and $\mathbf{p}_1 + \mathbf{a}_1 = (x_1 + a_1, y_1 + b_1, z_1 + c_1)$ can be represented parametrically as $\mathbf{p}(s) = \mathbf{p}_1 + s\ \mathbf{a}_1$, $0 \le s \le 1$. The line segment appears in the image at the points

$$u = (x_1 + a_1\ s) \cos\theta - (y_1 + b_1\ s) \sin\theta \qquad (4.4)$$
$$v = y_1 + b_1\ s$$

specifying the appearance of an edge as the 2D surface in aspect space $(u(s, \theta), v(s), \theta)$. This is the form of the 2D surface in aspect space shown in Figure 4.9(b), which corresponds to the appearance at all viewpoints of the bold edge in Figure 4.9(a).

The term *visibility structure* will be used to denote the structure in aspect space corresponding to the visibility of a particular model feature. The dimensionality of the visibility structure for a feature depends on both the dimensionality of the feature and the dimensionality of viewpoint space. For visual simplicity we have illustrated visibility structures using a 1D viewpoint space so that the visibility structure for edges is a 2D surface and the visibility structure for a face is a 3D volume. With two degrees of freedom in viewpoint, the dimensionality of the visibility structures for faces, edges and vertices each increase by 1.

A fundamental property of aspect space is that occlusion is equivalent to set subtraction in aspect space [Plan88]. Consider two faces and their corresponding visibility structures. A point that lies within the visibility structure for both faces is a single image point generated from both faces. Since only one face can be visible, the point is removed from the visibility structure for the face that is occluded. Thus, the exact set of visible points of a face from all viewing directions can be computed by performing set subtraction in aspect space. The visibility structure for a face is a volume bounded by algebraic surfaces. The intersection operations for two such structures can be done in closed form in a way similar to polyhedral intersection.

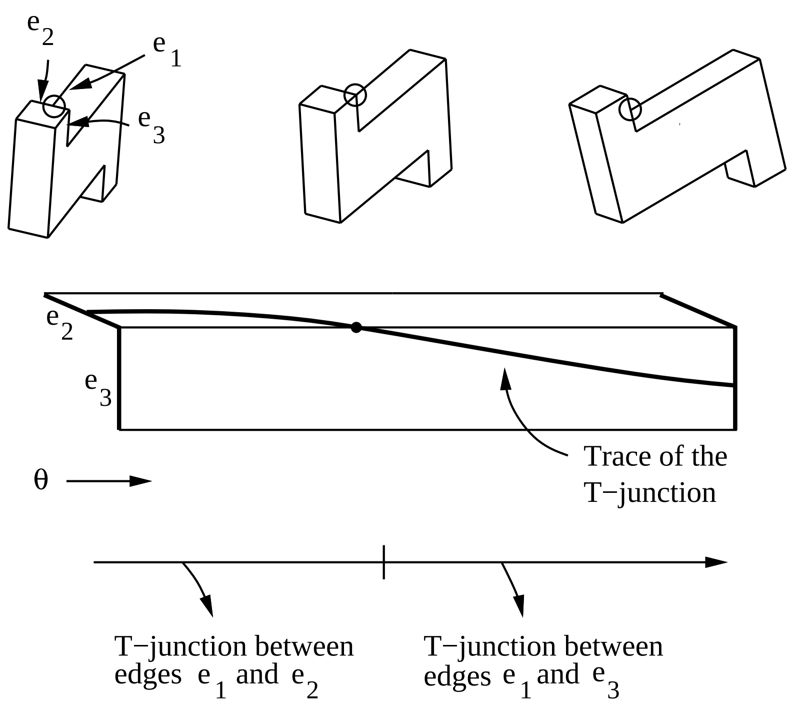


Figure 4.10. (a) The viewpoint changes from left to right. The T-junction of interest is circled. (b) The 2D surfaces in aspect space for edges e_2 and e_3. The bold curve on the surfaces is the trace of the T-junction between e_1 and edges e_2 and e_3. (c) A linked set of intervals along the viewpoint axis represents the edges that form the T-junction.

The asp is the explicit boundary model of the visibility structures in aspect space that corresponds to the visibility of each vertex, edge and face of a polyhedron [Plan88, Plan90]. The asp for a polyhedron quantifies the appearance of each face, taking into account the occlusion relationships between faces. The asp for a single planar face can be described exactly by computing the equations of its boundaries in aspect space. For example, the asp for face A in Figure 4.9(a) is the volume in aspect space that is bounded by the surfaces swept out by the edges around face A. The surface in Figure 4.9(b) is one of these boundaries that corresponds to the common edge of face A and face B. The asp was originally introduced as an intermediate structure for constructing the aspect graph. The construction of the asp is a well-defined procedure with known, tight bounds on the time and space requirements.

4.3.2. Explicit Representation of the Polyhedral Rim

The rim appearance representation is the organized collection of the visibility structures in aspect space corresponding to the rim edges of a polyhedron. The visibility structure for a rim edge is a section of the visibility structure for an edge corresponding to the viewpoints where the edge is on the rim. Figure 4.9(b) shows the visibility structure in aspect space for the bold edge in Figure 4.9(a). The shaded sections of the visibility structure are those sections where the edge is on the rim. A portion of the visibility structure is cut away at the viewpoints where the edge leaves the rim so that only the visibility of the edge as a part of the rim is represented.

The asp is the visibility structure that represents the exact appearance of a face as a function of viewpoint. The asp, however, does not make explicit the visual events that affect the rim. In contrast, there are three characteristics that we feel are necessary in a representation of the rim appearance:

- The visibility structures for rim edges must be connected together over viewpoints where the set of rim edges changes

- Only visual events that involve rim edges should be included

- The exact appearance of the rim must be encoded, including an explicit representation of T-junctions and contour splits and merges

The following paragraphs describe how the rim appearance representation meets these goals.

Each visibility structure encodes the exact visibility of a single rim edge for all viewpoints, and structures are connected together in aspect space where they are adjacent. This explicit connection allows visual events involving the visible rim (i.e., T-

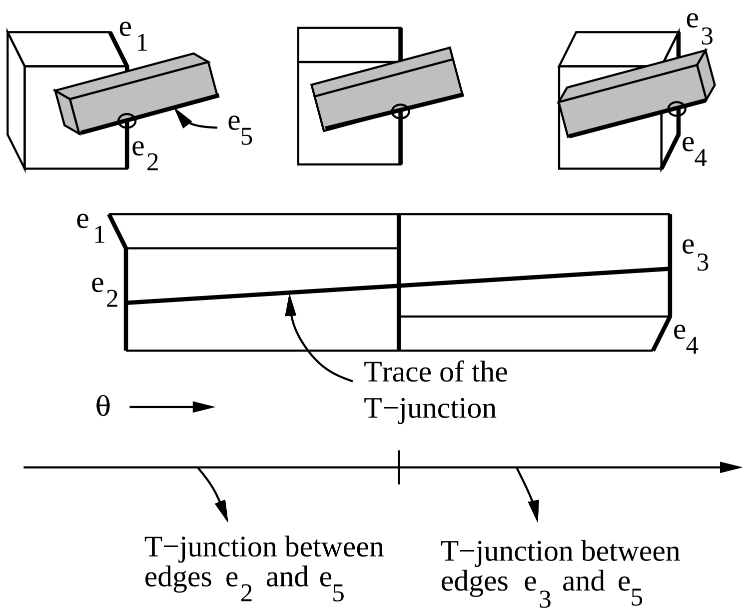


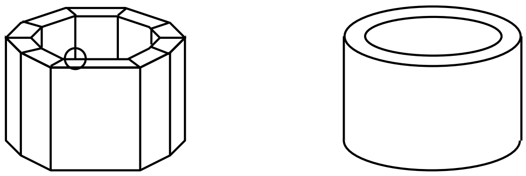
Figure 4.11. (a) The rim edges of a cube being occluded by a rectangular solid. (b) The 2D surfaces in aspect space for the rim edges of the cube. The line that spans the surfaces is the trace of the T-junction between e_5 and the rim edges of the cube. (c) A linked set of intervals along the viewpoint axis represents the edges that form the T-junction.

junctions) to be represented as piecewise objects. The visibility structures for rim edges can be adjacent in aspect space as a result of two geometric conditions. First, two edges that share a common vertex in the model are adjacent in aspect space. Visual events may cross the boundary between the two adjacent visibility structures. Figure 4.10 illustrates the geometry of two adjacent edges and their corresponding visibility structures in aspect space. The circled T-junction in Figure 4.10(a) occurs first between edges e_1 and e_2. As the viewing direction changes, the T-junction crosses the Edge-Vertex (EV) event where the common vertex between connected edges e_2 and e_3 projects onto e_1. In the right-most view, the T-junction occurs between edges e_1 and e_3. The geometry in aspect space is illustrated in Figure 4.10(b). The trace of the T-junction is shown as the bold curve that cuts across the two adjacent surfaces in aspect space. The point where the bold curve crosses the adjacency boundary is the EV-event between e_1, e_2 and e_3.

The second geometric condition that causes adjacency in aspect space is the appearance and disappearance of rim edges. When a rim edge disappears, another takes its place. In aspect space, this corresponds to an adjacency in the viewpoint dimension between the disappearing and appearing edges. Figure 4.11 illustrates this type of adjacency in aspect space as the set of rim edges changes with viewpoint. As viewpoint changes, the edges that participate in the circled T-junction change because the set of rim edges changes. Figure 4.11(b) shows how the visibility structures for the rim edges e_1 and e_2 are connected to the adjacent visibility structures for e_3 and e_4. Visibility structures for individual rim edges are connected together in aspect space as part of the same contour.

The visibility structure for an edge can be constructed so that it encodes only the visibility of the edge as part of the rim. Thus only visual events that involve rim edges are computed and stored. Figure 4.12(a) shows a cylindrical shape that is approximated by a non-convex polyhedron. The visual event circled in Figure 4.12(a) is part of the appearance of one of the faces of the model, and is explicitly represented in the asp. This feature will never be observable in an image, however, and hence is not represented in the rim appearance representation.

A cross-section of the rim appearance representation for a fixed viewpoint corresponds to the exact appearance of the occluding contour in the image plane. Unlike the asp, the rim appearance representation encodes the appearance of the visible occluding contour, not of individual faces. The visual events involving the rim throughout



(a) (b)

Figure 4.12. (a) A polyhedral approximation of a curved object. (b) The actual smooth object. The visual events like the one circled in (a) will never be produced by the object in (b). Such visual events are not represented in the rim appearance representation.

viewpoint space can be extracted and organized explicitly to define the T-junctions that occur in the projected model. In fact, constraints between pairs of these visual events can be formulated explicitly in terms of orientation, spatial relationship, and derivatives with respect to change in viewing direction. By representing the occluding contour as a function of viewpoint, the evolution of the appearance of the occluding contour over viewpoint can be extracted and summarized hierarchically from cross-sections and from visual event properties. Thus properties of the occluding contour such as curvature extrema and merging and splitting can be explicitly represented and organized.

Several snapshots of the appearance of the rim for a polyhedral model of a candlestick are shown in Figure 4.13. These views were generated by an implementation that constructs the rim appearance representation for one degree of freedom in viewpoint. The rim appearance representation encodes each of the T-junctions between the contours of the candlestick as a single structure. The visual events that affect the T-junction are stored explicitly. For example, the top sequence in Figure 4.13 shows the migration of the circled T-junction as viewpoint changes. The second view shows a triple-contour intersection (EEE-event) where two T-junctions temporarily coincide. The third view shows that the T-junction has evolved from an interaction between the base and top of the candlestick to the interaction between the base and the middle of the candlestick.

The bottom row of snapshots in Figure 4.13 illustrates the splitting and merging of an occluding contour. The occluding contour corresponding to the top section of the candlestick in the leftmost view is unbroken. A slight change in viewpoint causes the contour to break at a concave edge. Likewise, part of the contour corresponding to the middle section splits at a concave edge. In the third view, the contours have merged again into a single silhouette (except for the contour at the hole in the candlestick). The merging occurs at viewpoints where the T-junctions end (EV-events) and the rim edges are connected spatially into a single circuit.

4.3.3. A Construction Algorithm

The algorithm for constructing the rim appearance representation is similar to the algorithm for constructing the asp. The critical differences are (1) the connection of visibility structures in aspect space based on how the set of rim edges changes with respect to

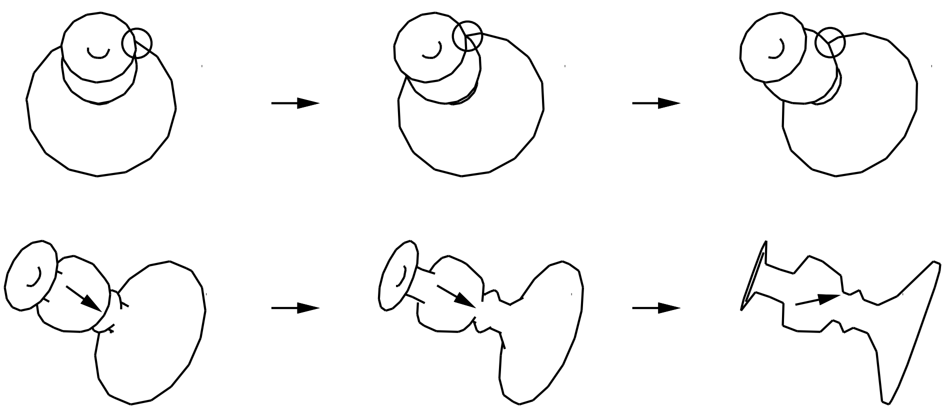


Figure 4.13. The rim appearance representation encodes the T-junctions and the contours in this model as a piecewise-continuous function of viewpoint.

viewpoint, and (2) the restriction of the visibility structure of an edge to the rim. The steps below outline an algorithm to construct the rim appearance representation:

(1) **Compute the visibility structure for all rim edges.** The local definition of the rim restricts the regions in aspect space where an edge is visible as a rim edge. The two disjoint regions in aspect space where each edge is locally defined to be on the rim are constructed.

(2) **Determine EE-events between rim edges.** Since only rim edges can cause visual events (of interest) with rim edges, the edges are tested to determine those pairs of edges that form a T-junction from some set of viewpoints. For such edges, their visibility structures are constructed and then modified as follows:

> (a) Given the visibility structure for a single rim edge, subtract the visibility structures for all other intersecting rim edges. The resulting visibility structure is the appearance of the rim edge including the boundaries in aspect space corresponding to T-junctions formed with other rim edges. The algebraic boundaries generated by this process are specified by the equations for the EV- and EEE-events [Plan88].
>
> (b) The final visibility structure for a single edge must be divided into a set of potentially disjoint volumes. This division is based on the global visibility of the edge. The global visibility test must be made to determine if the edge is fully visible or completely occluded. However, since visibility can only change at rim edge boundaries, this global test needs to be performed only once for the entire visibility structure for an edge. The result of this test can be propagated to adjacent sections of the visibility structure to determine which pieces need to be cut away as a result of total occlusion. Note that the fact that the visibility can only change at EE-events implies that the property of being totally occluded or totally visible must remain constant across a connected section of the visibility structure.

(3) **Connect the visibility structures for individual rim edges across both spatial and viewpoint dimensions in aspect space.** Spatial connections are specified by the connectivity information in the polyhedral model. Thus the visibility structures for two

spatially-connected rim edges can share the curve in aspect space defined by the visibility of their common vertex. The visibility structures for two distinct edges are connected in viewpoint space at those viewpoints where one edge leaves the rim and the other edge becomes part of the rim. This “sewing” step is necessary because given this information, a single T-junction can be described across viewpoint as a piecewise-connected curve or surface that is independent of its constituent edges. That is, the events where edges disappear from the rim can be ignored and the connectivity information can be used to describe the T-junctions across these boundaries.

4.3.4. Complexity Analysis

The complexity of constructing the rim appearance representation is bounded by the number of visual events that affect the appearance of the rim. The number of vertices in the visibility structure of the rim appearance representation in aspect space determines the construction time and space complexity. Under orthographic projection with two degrees of freedom in viewpoint, each vertex in the rim appearance representation is generated by the apparent intersection of four object edges. Thus, in the worst case, the rim appearance representation can be constructed in space $O(\mathbf{n}^4)$ for a polyhedron with $\mathbf{n}$ faces. Since the algorithm must compute the intersection of the visibility structure for each rim edge with every other, the construction time is bounded by $O(\mathbf{n}^5)$.

These complexity bounds are the same as those for constructing the asp [Plan88]. As with the asp, pathological polyhedra such as picket fences and grids can achieve the worst-case behavior. The rim appearance representation, however, has a much better average case behavior because of the elimination of many of the visual events that occur in polyhedra but are not related to the rim or the occluding contour (as illustrated in Figure 4.12).

4.4. Approximating the Rim Appearance

The exact appearance of a rim edge can be a complex structure in aspect space. In general, the 1D boundaries in aspect space of the visibility structure for a rim edge correspond to EEE-events. In the worst case, the visibility structure for a rim edge can be a non-connected region bounded by many EEE-events. This complexity motivates consideration of an approximate representation that sacrifices exactness for simplicity.

This section presents an approximate representation of the rim appearance that is constructed by ignoring the EEE-event. Intuitively, this means that the appearance of a rim edge is represented with T-junctions only, and a T-junction between two edges must be marked as either completely visible or completely occluded. This is necessary because the EEE-event is the "transition" event that occurs when a T-junction (EE-event) either appears or becomes occluded.

The following subsections describe the geometry of the approximate rim appearance representation, the algorithm for computing this structure, and an analysis of the construction cost in time and space. The complexity results are asymptotically better than for the exact rim appearance representation because EEE-events are not computed.

4.4.1. Eliminating the EEE-Event

The visibility structure in aspect space for a rim edge is altered by both EE-events and EEE-events to produce a bounded set of regions. This set of regions is the exact appearance of the rim edge as it is occluded by other parts of the model. The EEE-event is potentially the most numerous and costly to compute. A large savings in space and time can be traded for exacteness by eliminating the EEE-event.

Consider a partially occluded rim edge that forms a T-junction. This T-junction may be visible or may be occluded by some other part of the model. The viewpoints where the T-junction becomes occluded are just those viewpoints where EEE-events occur. The basis for our approximate scheme is to eliminate EE-events based on a visibility test. Rather than splitting an EE-event boundary at the place where the EEE-event occurs, we either retain the *entire* EE-event, or *completely discard* it. The potential for EEE-events (the intersection in aspect space of two different EE-events for a rim edge) is not computed. Clearly this scheme is approximate, since it is possible to discard T-junctions that are partially visible. It is also possible that a visible T-junction will actually be mostly occluded.

The primary difference between the computation of the approximate rim appearance and the exact rim appearance is that global visibility tests must be made instead of the calculation of EEE-events. The cost of computing a complete event description is traded for the cost of performing a global visibility test using the model for each EE-event. The time necessary to perform a visibility test for every EE-event is O($\mathbf{e}^3$) where $\mathbf{e}$ is the number of model edges. This results from the $\mathbf{e}^2$ possible EE-events that each must each be tested in O($\mathbf{e}$) time per test. The construction time of the rim appearance representation is therefore O($\mathbf{e}^5$) because the visibility structure of each rim edge must be intersected with every other.

The primary tradeoff in what is represented is space versus exactness. The approximate representation is much smaller without the EEE-events, although it is inexact. The quality of the approximation depends on many factors, including the persistence of the rim edges in the model as well as the overall visual complexity of the object. This makes it difficult to quantify. We argue, however, that the closer the polyhedron approximates a smooth shape, the more advantage the approximation has over the exact representation of appearance. This is because polyhedra that are "smooth" have rim edges that persist in viewpoint space for only a small range of views. Over this small set of views, the quantitative description of the EE-events that occur is quite stable [Burn90]. As the polyhedron becomes a better and better approximation of a smooth object, the cost of maintaining exact information is much higher than the gain in accuracy over this approximate method.

4.4.2. A Construction Algorithm

The algorithm to construct the approximate rim appearance information relies on finding EE-events between rim edges and then testing these EE-events for visibility (whether or not they are, for the most part, visible). The visibility test is used to determine whether an EE-event should be retained or discarded. Occluded EE-events are those that are discarded. Outlined below are the major steps of the construction algorithm:

(1) **Compute the initial visibility structure for all rim edges.** The local definition of the rim restricts the regions in aspect space where an edge is visible as a rim edge. This is the same as the first step in the exact algorithm.

(2) **Find EE-events between rim edges.** Rim edges are tested to determine those pairs of edges that form a T-junction from some set of viewpoints. For each EE-event, a visibility test is performed:

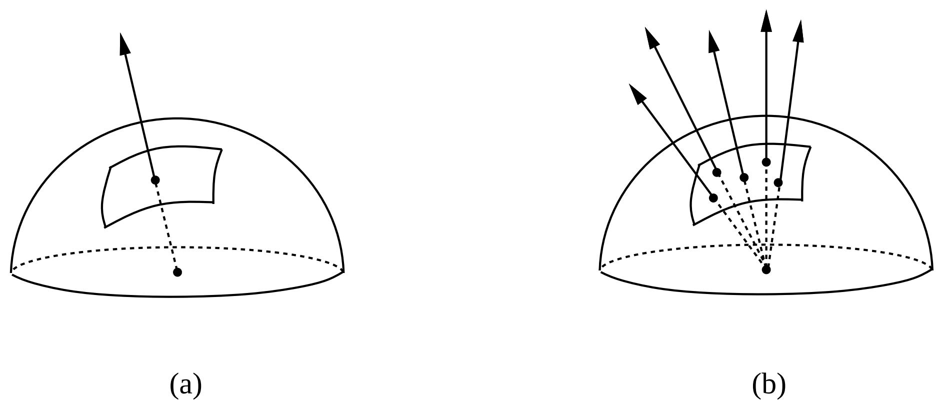


Figure 4.14. (a) The visibility of an EE-event (T-junction) can be approximated by the visibility at a single viewpoint. (b) Several sampled viewpoints within the EE-event give an indication of the degree of partial occlusion.

(a) If the EE-event is determined to be occluded, discard it.

(b) If the EE-event is determined to be visible, retain it.

(3) **Determine final rim edge visibility.** Determine the visibility of each rim edge in the region of viewpoint space where it is not involved in any EE-event. The geometry of an EE-event on the view sphere is a convex spherical polygon and hence this test is simple.

The visibility test is the critical part of this algorithm. A simple test for each EE-event can be done by finding the central viewpoint within the EE-event and determining if the EE-event is visible from that viewpoint. Figure 4.14(a) illustrates the central viewpoint of an EE-event on the view sphere. This EE-event may be completely occluded by some other part of the model. A single visibility test for an EE-event cannot determine the degree of partial occlusion. Several tests across the EE-event could more accurately determine its visibility. Figure 4.14(b) shows how several viewpoints within the EE-event could be used to determine the final fate (discard or retain) of an EE-event.

The specific issues involved with approximating the rim appearance are related to the broader problems of scale and model accuracy in all computer vision work. The approximation presented here of the exact rim appearance is an approximation in viewpoint space, where the complexity is reduced by approximate visibility criteria. There are other possible approaches to these problems, each with its own various trade-offs. This is an area of active research [Ponc90a, Bowy91, Dick91].

4.4.3. Complexity Analysis

The complexity of constructing the approximate rim appearance representation is bounded by the number of EE-events that affect the appearance of the rim. The number of EE-events determines the construction time and space complexity. Unlike the exact algorithm, EEE-events are not computed. Under orthographic projection with two degrees of freedom in viewpoint, in the worst case, the approximate rim appearance representation will require $O(\mathbf{n}^2)$ space for a polyhedron with **n** faces. Since the algorithm must perform a visibility test for each EE-event, the construction time is bounded by $O(\mathbf{n}^3)$. These complexity bounds are better than those for computing visual events exactly. Pathological polyhedra such as picket fences and grids can still achieve the worst-case behavior.

4.5. Implementation Results and Analysis

The construction algorithm for the rim appearance representation has been implemented for 3D aspect space. The implementation is coded in C and uses an X-windows interface for object display. Table 4.1 shows results generated by the program for the polyhedral models in Figure 3.12. The number of visual events stored in the rim appearance representation is compared to the number of visual events computed for the display algorithm in Chapter 3. The storage requirement for a single visual event is approximately 20 bytes. The size of the aspect graph for the models is not shown in the table because the aspect graph is always as large or larger than the event data used for exact hidden-line display.

Model	Faces	Total Events	Rim Events	Total Size (Kbytes)	Rim Size (Kbytes)
Chain link	72	996	220	24	6
Layers of squares	48	2365	544	28	8
Small grid	41	1674	435	36	12
Candlestick	288	3109	715	72	17
Candlestick and sphere	416	5270	1264	120	31
Three spheres	384	3733	970	88	24
Interlocked tori	512	5135	1294	104	30
Torus and candlestick	544	5030	1308	120	32
Spring	1800	39954	8790	868	220
Large grid	449	9776	2738	216	68
Torus inside torus	512	4395	1010	104	25

Table 4.1 The asp and the rim appearance representation were created for the polyhedral models above. The sixth column shows that the number of visual events computed for each of the models is reduced by 75% when computing the visual events involving only the rim edges. The visual events were generated under orthographic projection with one degree of freedom in viewpoint.

The columns in Table 4.1 show the total number of visual events in the asp and the number of these events that are rim events. Column 6 shows the percentage of the total events that are rim events. The rightmost two columns report the average number of events per edge in the asp and the average number of events per edge in the rim appearance representation. As the sixth column shows, there was approximately a 75% reduction in the number of visual events to be considered in the rim appearance representation: for example, for the candlestick model, the average number of visual events per edge decreased from 5.5 in the asp to 1.5 in the rim appearance representation. These results indicate that the rim appearance representation saves significant time and space over the asp (and hence the aspect graph as well) while preserving the completeness of the exact appearance of the occluding contour at all viewpoints.

4.6. Concluding Remarks

This chapter has described a novel, viewer-centered model of the occluding contour of 3D objects called the rim appearance representation. The exact appearance of the rim edges of a polyhedral approximation of a smooth object is encoded by the rim appearance representation. The appearance of the polyhedral rim and the features of self-occlusion that affect its appearance correspond to the contour events generated by smooth objects. The linear edges of polyhedra make it possible to represent both T-junctions and occluding contour events as explicit objects that are organized in terms of sets of interacting rim edges. Representing these features explicitly is difficult in general because of the continuously changing rim points of smooth surfaces.

A viewer-centered model of the occluding contour of 3D shape is important for model-based vision. The occluding contour of 3D shape is related to 3D surface properties, and the features of self-occlusion provide strong constraints for recognition. Features such as T-junctions and contour events are intrinsic to projected shape, and can be represented and used for indexing and matching in a model-based system. The relationships between occluding contour features can further constrain the matching process. The viewer-centered approach to modeling 3D shapes makes the changes in features with respect to viewpoint explicit. These changes can be used in a dynamic context where image features are observed changing over time, or where matching methods must iteratively refine an estimation of viewpoint [Lowe89]. In addition, an explicit model that includes features of self-occlusion for solid shape makes the prediction of the appearance of the model a faster process that can speed up model matching [Basr88].

Although the rim appearance representation can be large for worst-case examples, the average case requires much less time and space. Consequently, this representation is much smaller on average than other representations such as aspect graphs and the asp. The organization of individual rim edges into contour-level events gives a natural abstraction and useful structure to the visual events that affect the rim. Finally, it should be noted that this model of the occluding contour is intended to be used in conjunction with other available surface and geometric information. The integration of this 3D viewer-centered approach with other geometric features should provide a strong foundation for more sophisticated, shape-based approaches to 3D vision.

Chapter 5

Viewpoint from Occluding Contour

The viewer-centered rim appearance representation stores the features of the occluding contour for polyhedral models. This chapter considers the problem of how to constrain the set of viewpoints from which a 3D model will project to an observed set of occluding contour features using the rim appearance representation. This approach relies on precomputed features of the occluding contour stored in the rim appearance representation. These features are organized into a structure making inter-feature relationships and dynamic feature changes explicit. It is assumed that geometric information about the models is known, occluding contours can be detected in the image, and information such as surface normals or texture is not available.

The problem of constraining viewpoint for a particular model is often considered a subproblem of model-based 3D object recognition [Lowe87]. In general, the model-based approach is to select a model **M** and the corresponding viewpoint **V** that will produce a projection that best matches the image data. For each model, the best viewpoint **V** selected from a space of all possible viewpoints is computed. Recognition is the selection of the best model,viewpoint pair, i.e., the pair (**M**, **V**) with the highest degree of match. Thus a fundamental problem under this paradigm is *viewpoint determination*, i.e., the computation of the model viewpoint that best matches the image for a given model. This is essentially equivalent to *pose determination*, where an object's pose also describes the transformation that relates the object position to the camera position in world coordinates.

The approach developed in this chapter can be distinguished from previous methods in two respects: (1) the kinds of features that are explicitly represented, and (2) the type of model-image correspondences that are made. First, the appearance of the occluding contour, including the formation and persistence of T-junctions, is represented. There is strong information available in the occluding contour, but the difficulty in adequately and explicitly describing it has prevented its use in the past. Second, the shape and topology of the occluding contour is usually stable over a range of viewpoints but is not generated by a fixed, intrinsic feature of the shape. For example, a T-junction produced by the projection of a smooth shape persists over an open set in the space of viewpoints, yet the 3D points that project to the T-junction are not fixed over that set. Consequently, a correspondence between a T-junction detected in an image and a model T-junction is different from a point-point correspondence; a T-junction correspondence defines a connected set of viewpoints where the T-junction can occur rather than a single transformation to bring the model features into exact correspondence with the image. This notion of correspondence is more qualitative, producing a small, constrained region of viewpoints rather than a single viewpoint that generates the image features. This implies a two-step procedure for viewpoint determination: (1) finding a constrained region of viewpoints, and (2) find a single viewpoint within the region as a solution.

There are certain advantages in making use of features of the occluding contour for viewpoint determination. The occluding contour provides strong information for viewpoint constraints and for 3D object recognition [Koen90, Rich88], although until now there has been little work to incorporate this information into a model-based approach. The depth ordering of surfaces relative to the viewer and qualitative pose information can be inferred from T-junctions and their relative orientations. The curvature of the occluding contour is directly related to the 3D surface generating it [Koen90].

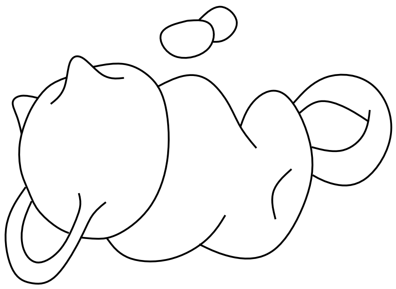


Figure 5.1. The occluding contour of a 3D non-convex object is rich in visual cues that strongly constrain viewpoint. This figure was taken from [Koen87].

The relationship between sets of occluding contour features generated by arbitrary non-convex 3D shape strongly constrains the possible viewpoints that can generate those features. For example, the projected 3D object in Figure 5.1 is rich with visual cues such as T-junctions, ending contours, concave and convex arcs, and inflections. These features independently and in relationship to each other constrain the possible views that can generate the given projection.

Initially solving for a constrained viewpoint region using the occluding contour is necessary for many other methods for object recognition because they assume a given, approximate solution. In the case of iterative methods [Lowe87], a critically necessary starting viewpoint can be obtained from a constrained viewpoint set. Parameter space methods [Thom87] that can avoid a costly search of the entire space of transformations become much more efficient. The search of an interpretation tree [Grim90] is more efficient in space and time when there are global constraints on the possible solutions. Aspect graph methods [Bowy89, Krie90, Ikeu87] must address the problem of how to select a few aspects to test from a large number of potential aspects.

There are several important advantages to using the rim appearance representation over the aspect graph. The aspect graph is constructed from nodes representing regions of viewpoint space and arcs representing adjacencies in viewpoint space. Viewpoint space is divided so as to preserve *constant aspect* within each connected region. Two views are in the same aspect when the features projected from those views are qualitatively the same. Aspect graph approaches have not yet solved the nontrivial problem of selecting the appropriate aspect given a set of detected features. We solve this problem in part by representing individual features across aspect. The rim appearance representation makes the changes in individual features and in feature sets across adjacent aspects more explicit than in the aspect graph. Also, only occluding contour features are represented. If it is assumed that the polyhedral model is an approximation of a smooth shape, many of the polyhedral events represented in the aspect graph are not observable in the projection of a smooth shape.

This chapter describes an approach to solving the model-based viewpoint determination problem using shape features of the occluding contour. In particular, the inflections and T-junctions that arise from self-occlusion and non-convexity are used as features to first determine a region of viewpoints that matches the image. These contour features are precomputed and, unlike the aspect graph, are organized into a structure that makes inter-feature relationships and dynamic feature changes explicit. The precomputed geometry of the occluding contour, geometry that is dependent on viewpoint, is related to image features. This relationship globally constrains viewpoint and provides a direct model-to-data correspondence for features of the occluding contour. Included in this framework is a representation of the dynamic evolution of the occluding contour as a function of viewpoint. This dynamic information, as a part of the object model, provides another way to extract quantitative information about viewpoint from image changes over time. After this qualitative solution, resulting in a viewpoint region, is obtained, a second procedure is used to precisely align the model and image features so as to determine a final single viewpoint.

Polyhedral object models are used, and the behavior of the occluding contours of these models is precomputed and organized for searching and matching. Although the occluding contour of a polyhedral model does not change *continuously* with viewpoint, it changes discretely and provides an approximation of the smooth occluding contour in the polyhedral domain. Section 5.1 specifies how features of the occluding contour appear in the image plane and identifies feature properties that constrain viewpoint. The feature selection and search issues involving the rim appearance representation are presented in Section 5.2. Section 5.3 gives implementation results from a prototype system that constructs and organizes the rim appearance representation for polyhedra and then relates this information to projected contour features. Results include information about the number and the persistence of precomputed model contour features, as well as an algorithm to select model features in order to account for the occluding contour features in images. Section 5.4 summarizes the results, and discusses a way to integrate the rim appearance representation with the interpretation tree paradigm for a more constrained model-based strategy.

5.1. Contour Geometry and Organization

The features of the occluding contour that are precomputed for a particular model are defined by viewing direction, self-occlusion from projection, and the curvature of the surface at the rim. Thus the association of a feature in an image to a model feature implicitly includes a set of constraints on viewpoint. These constraints are derived from the geometry that produces the contour feature under projection.

This section focuses on the following issues: the geometry of contour features, the representational organization of these features, and the viewpoint constraints resulting from model-image correspondences. The problem of model-image correspondence is treated as a *contour-level* correspondence rather than an exact edge-edge correspondence. This is based on the organization of contour features as a function of viewpoint.

Smooth opaque shapes without surface discontinuities generate only T-junctions, smooth occluding contours, and contour terminals in the image plane [Mali87]. It is assumed here that the polyhedral model is an approximation of a smooth shape. It can then be assumed that none of the polyhedral edges are true surface discontinuities. In general, surface discontinuities can be treated without difficulty, but for simplicity the discussion here is restricted to smooth surfaces. In this case the only stable, or transverse, junctions in the smooth occluding contour are T-junctions and contour terminals. Triple-contour (or higher order) intersections can occur, but only from a 1D or 0D set of viewpoints. Consequently, any perturbation of viewpoint within an open disc on the view sphere will cause a triple-contour feature to disappear. Accordingly, the features described in detail here are the T-junction, caused by the intersection in the image plane of two non-adjacent rim edges, and the contour terminal, caused by concave edges. Both of these features persist in general over a 2D set of viewpoints.

5.1.1. T-junctions

The EE-event is represented in the rim appearance representation as a hypersurface in aspect space. A set of EE-event hypersurfaces that are adjacent in aspect space establishes a piecewise representation for a single T-junction between two occluding contours. This piecewise-connected set of hypersurfaces approximates a single contour-contour event (CC-event). The CC-event is the T-junction that is produced by the projection of two smooth surfaces. The difficulty in explicitly representing T-junctions for smooth surfaces is the numerical complexity of the locus of surface points that define the viewing directions where T-junctions occur [Krie90]. For polyhedra, the EE-events assembled into piecewise CC-event structures provide an approximation of the smooth T-junction geometry.

There are three important geometric quantities for a T-junction that are represented in the rim appearance representation: the image location of the apparent intersection

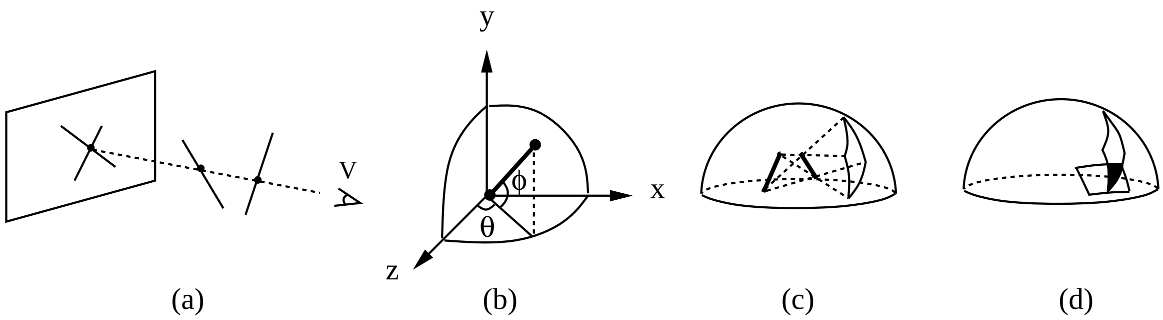


Figure 5.2. (a) The T-junction formed by a pair of model edges. Both edges are restricted to the rim. (b) The orthographic viewing model. (c) A T-junction persists for a bounded viewing region. (d) Two T-junctions co-occur at the viewpoints where their patches overlap on the viewing sphere.

point, the orientation of the T-junction, and the viewpoints where a T-junction is created or annihilated. These properties are directly computable from the geometry of the two edges that form a T-junction; the piecewise connection of EE-events represents the change in the T-junction as the rim changes. The orientation and location in the image plane are immediately available from the geometry of the edges that form it. The set of viewpoints where the T-junction is created or annihilated occur where no continuation of the T-junction to other rim edges is possible. The equations for the EE-event specify the image location, orientation and the viewpoints where the event is created or annihilated.

The geometry of the interaction of two rim edges that form a T-junction is described in part by a region of visibility on the view sphere, as shown in Figure 5.2(c). Figure 5.2(a) shows for simplicity only two isolated edges although, under the definition of the polyhedral rim, edges forming a T-junction are constrained by the visibility of the faces that meet to form them. By the definition of the polyhedral rim, a particular edge can only form T-junctions for viewpoints where that edge is on the rim. The image coordinates of a projected T-junction for two edges are represented directly as a function of the viewing direction **V** and the endpoints of the segments [Plan88]. As shown in Figure 5.2(b), a viewpoint is modeled as a point (θ, ϕ) on the unit sphere. The orthographic projective transformation for this sphere of viewpoints is rotation by (θ, ϕ) and then orthographic projection in the direction of the z-axis.

There are two conditions that govern the observability of an EE-event between two rim edges. First, the local rim conditions must be satisfied: each edge must be on the rim and must project onto one another in the image plane. Second, the T-junction must be globally visible. Given only the local conditions, an EE-event may be occluded by another part of the surface. For example, let the set **S** be the set of viewpoints from which a given T-junction is observable in the image plane. Under the local conditions stated above, **S** forms a convex, connected set on the view sphere. The equations defining the boundaries of the set **S** in this case are functions of the endpoints of the edges and the normal to the faces meeting at each edge. These boundaries are sections of great circles so that **S** is a spherical quadrilateral.

When global occlusion modifies the visibility of the T-junction, **S** may no longer be connected. This can be visualized intuitively in the case where an EE-event (T-junction) is partially obscured by a grid. The set of viewpoints where the two edges project to the T-junction is split into disjoint regions where the T-junction is visible. The visibility boundaries caused by the grid are specified by the EEE-event [Plan88]. In $\mathbb{R}^3$, the locus of points where three lines project to a single point is a ruled quadric surface [Gigu90]. The intersection of the ruled surface for three edges with the view sphere forms a polynomial (quadratic) curve with coefficients that are a function of the endpoints of the three segments. Thus **S** can be a non-connected region on the view sphere where each connected subregion is bounded by a combination of great circles and quadratic curves generated by EEE-events [Plan90].

A set of T-junctions (rather than just one T-junction) can be produced by a non-convex shape at a single viewpoint. The geometric relationships between simultaneously visible T-junctions is of interest because of the additional viewpoint constraints that they provide. These relationships can be computed using the geometry of the T-junction patch boundaries. Two or more T-junctions that can simultaneously occur in an image define a strictly higher constrained region of viewpoints, i.e., the region defined by the intersection of the patches for each individual T-junction. Conversely, two T-junctions co-occur at the same viewpoint when their respective patches form a non-empty intersection on the view sphere. The region of intersection is the set of viewpoints where the edges are on the rim and the two pairs of edges will each project to a T-junction in the image. Figure 5.2(d) shows two patchs of viewpoints on the view sphere formed by two pairs of edges, and the intersection of those two patches where the T-junctions co-occur.

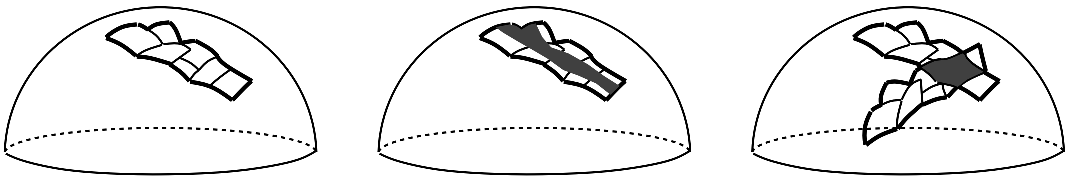
The presentation above explains the well-defined geometric structure for the set of hypersurfaces in aspect space that collectively approximate a CC-event. A CC-event is computed directly from a polyhedral model for the purpose of predicting how the occluding contour of the model will appear in the image as a function of viewpoint. An hypothesized correspondence between a CC-event computed from a model and a T-junction in the image produces a set of viewpoint constraints. These constraints are of three forms: CC-event persistence constraints, geometric constraints, and co-occurring feature constraints.

The CC-event persistence constraint eliminates viewpoints where the CC-event cannot occur. That is, if it is hypothesized that a particular image T-junction has been produced by a specific model CC-event, then the viewpoint that created the image projection must lie within the set of viewpoints that produce the CC-event. This set of viewpoints for a CC-event in the model is well-defined with exact boundaries.

Geometric constraints are derived from an assessment of how well the measured characteristics of an image T-junction matches a CC-event in the model. For example, the stem and the T of a detected T-junction meet together at a particular angle. The viewpoints bounding the CC-event are further constrained by boundaries around the viewpoints that will project to a T-junction that is approximately equal to the measured angle of the detected T-junction.

Co-occurring feature constraints are constraints between pairs of CC-events. An hypothesized correspondence involving one CC-event must also include evidence for any co-occurring CC-events. The set of viewpoints where two CC-events co-occur is in general much more restrictive than the set of viewpoints where a single CC-event occurs. Co-occurring constraints are simply persistence constraints involving a pair of CC-events. Specific geometric conditions describing the relative position and orientation of two CC-events can also constrain the set of viewpoints that may produce a pair of image features.

Each of these three types of constraints (CC-event persistence, geometric, and co-occurring) are illustrated in Figure 5.3. The figure shows a set of EE-events on the view sphere that are joined together to form a single CC-event. In Figure 5.3(a), the outer boundary of the union of EE-event boundaries is the CC-event boundary. Viewpoints



(a) (b) (c)

Figure 5.3. A CC-event is a piecewise collection of EE-events. (a) The persistence constraint: the outermost boundary (in bold) defines the CC-event region and delimits viewpoints where a single T-junction is visible. (b) The geometric constraint: the shaded viewpoints within the CC-event region are those viewpoints where the T-junction satisfies a detected geometric condition in the image such as a particular measured angle or orientation. (c) Co-occurring feature constraints: the shaded viewpoints are those viewpoints where two spatially-distinct CC-events co-occur in the image.

within this bounded region will produce the CC-event. A subset of the viewpoints within this CC-event region will satisfy specific geometric requirements from the observed image feature. Figure 5.3(b) illustrates how a subset of viewpoints satisfy specific geometric measurements such as relative orientation and angle of a T-junction. Viewpoints where two CC-events co-occur are computed exactly as an intersection of viewpoint regions on the view sphere. The shaded region in Figure 5.3(c) is the set of viewpoints where two model T-junctions will co-occur.

The edges for part of a CC-event can be aligned with an image T-junction for any of the viewing directions within the bounded region where the CC-event occurs. For a specific viewpoint, the rotation about the optical axis and translation in the image to bring the CC-event into correspondence with the image T-junction is well-defined. This is a result of the fact that a T-junction is *oriented*, with the stem of the T-junction being on the unoccluded side of the other edge. Figure 5.4 shows two model edges being aligned with an image T-junction. The edges are rotated by the viewpoint (θ, ϕ) and then projected to the image plane. The rotation α about the optical axis orients the edges

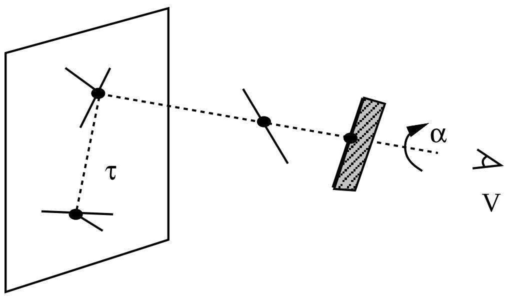
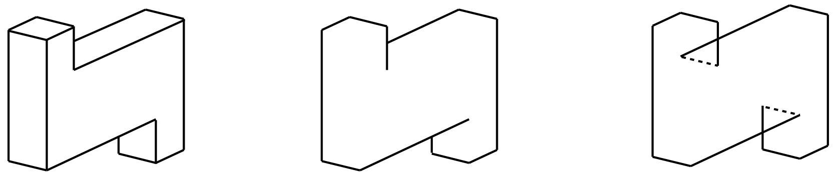


Figure 5.4. The T-junction formed by a pair of model edges is aligned with an image T-junction by a rotation α about the optical axis and an image plane translation τ to align the junction point.

appropriately. The translation τ in the image plane completes the alignment.

To summarize, a model-image correspondence constrains viewpoint since a CC-event exists over a bounded set of viewpoints. There are also geometric constraints such as the measured angle of a detected T-junction. Since a CC-event is represented as a piecewise set of adjacent EE-event hypersurfaces in aspect space, the exact geometry for each piece of the CC-event is available. It should be noted here that the viewpoints within a single EE-event that best match the stem and occluding edge of the image junction are difficult to compute. In general, there is a locus of viewpoints within an EE-event patch that can produce a T-junction that exactly matches an image T-junction. Despite this, the variation in the geometry of a T-junction formed by two edges depends on the size of the viewpoint patch where it occurs [Burn90]. Consequently, the geometry of a T-junction persisting over a small region of viewpoint space changes very little over that region, and can be closely approximated by the geometry at a single viewpoint within the region.



(a) (b) (c)

Figure 5.5. (a) A polyhedral model. (b) The occluding contour is broken by a combination of T-junctions and contour terminals. (c) The dotted edges are concave, and cause the contour terminals.

5.1.2. Contour Terminals

The occluding contours of smooth, transparent shapes are always closed curves that may contain non-smooth singularities such as cusps. These curves may overlap in the image plane and can have self-intersections. For closed, transparent polyhedra, the occluding contour is always a set of closed, linked edges that can overlap and can self-intersect. For opaque objects, on the other hand, self-occlusion (the overlapping of two different closed contour loops in the image plane) causes a contour to split into disjoint parts, or to merge into a larger piece. A "swallowtail" breaks the occluding contour at the T-junction that is created by the self-occlusion of an opaque surface [Koen90].

The rim appearance representation computes all of the visual events that affect the visibility of rim edges. These events include T-junctions that cause the occluding contour to split and merge, as well as the EV-events and EEE-events where T-junctions begin and end. The topological behavior of the occluding contour is computed and stored by organizing all of these events as a function of viewpoint. The topological characteristics are precomputed, and the exact appearance of the occluding contour for any viewpoint is directly available from the rim appearance representation.

The high-level contour representation that is constructed connects rim edges together that are spatially adjacent. In addition, the viewpoints where merges and splits occur as a result of visual events are explicitly computed so that the higher-level behavior of the occluding contour as a function of viewpoint is encoded. For example, the rim edges for a convex polyhedron make up a single loop of edges. Since there are no EV-events and EEE-events that cause the rim of a convex solid to split and merge, the behavior of the occluding contour is constructed as an unbroken loop of edges that never splits or merges. This higher-level structure provides a representation of the topological features of the occluding contour that is independent of the constituent edges.

The geometry of contour terminals is a key part of the change in the topology of the occluding contour across viewpoint [Koen90]. Contours end in polyhedra at *concave* edges only, i.e., at edges where the measure of the exterior angle between the faces meeting at the edge is less than π. Concave edges can be excluded from the rim since a concave edge can never occlude anything behind it. When concave edges are removed from the rim set, the rim can form a set of unconnected edges. For example, Figure 5.5(a) shows a polyhedron from a viewpoint where the set of visible rim edges is not connected in $\mathbb{R}^3$. The discontinuity occurs at the dashed concave edges, shown in Figure 5.5(c). The broken contour that results under projection, shown in Figure 5.5(b), consists of two contour sections. Each contour corresponds to part of the disjoint rim on the model. A concave edge causes the contour to end at either a T-junction (due to self-occlusion) or at a contour terminal.

The viewpoints where a contour terminal disappears (or a T-junction disappears) form visibility boundaries on the view sphere. These boundaries mark a change in the topolgy of the contour; the topology of the projected contour is different for each region. The set of viewpoints where the contour breaks at terminals and T-junctions is important because those viewpoints bound regions where the contour topology changes. Over regions of constant contour topology, points such as contour curvature extrema can be computed, and the spatial relationships of those points to the projected T-junctions and contour terminals can be computed. This qualitative description of the contour provides a coarse description of the contour in terms of qualitative connectivity and curvature information. At the same time, exact contour appearance is still maintained as sets of polyhedral edges and their interactions.

The rim appearance representation organizes polyhedral contour events. In the smooth case, the locus of viewpoints where visual events occur can be specified as solutions to algebraic equations. The solutions to such equations, however, are numerically complex [Krie90]. The polyhedral approximation eliminates the numerical complexity and provides a piecewise organization that preserves many of the interesting and detectable occluding contour features.

In summary, the rim appearance representation explicitly represents the visual events that cause a change in the topology of the occluding contour. The interaction of occluding contours in the form of a T-junction is represented as a piecewise-continuous structure that is connected across the changes in the participating rim edges. The locus of viewpoints that corresponds to these visual events in the smooth case is numerically complex and difficult to make explicit. The rim appearance representation avoids these problems by using the discrete rim of a polyhedral approximation to the real, smooth shape. The higher-level contour is represented as a set of edges that splits and merges as a function of viewpoint. This higher-level representation is important because its topology corresponds to the topology of the occluding contour of the smooth object that the polyhedron models. The splitting and merging of the occluding contour as well as the T-junctions that bound these splits and merges correspond to the dynamic behavior of the occluding contour of a smooth object. The topological behavior of the occluding contour and the explicit quantitative representation of the appearance of the occluding contour can then be used together as the basis for a model-based recognition system.

5.2. Feature Selection and Refinement

The rim appearance representation stores the appearance of the occluding contour for a complete set of viewpoints. This representation includes the geometry of T-junctions, contour terminals, and other features generated from self-occlusion. The organization of occluding contour information and the viewpoint constraints associated with each individual occluding contour feature is the basis for obtaining a viewpoint solution in the form of a small bounded set of viewpoints. This section presents an algorithm that has been implemented for using T-junctions and contour fragments in order to solve for a set of viewpoints matching an image. The problem is formulated as a search problem through the rim appearance representation for a given model. This search problem consists of an indexing phase, where an initial correspondence between model and image features results in a restricted set of viewpoints, and a search phase where the results of the indexing phase are refined by using the piecewise structure of the CC-event in the rim appearance representation. The connections linking individual EE-events over viewpoint are traversed to refine hypothesized correspondences.

5.2.1. CC-Event Selection

Contour features can be organized by characteristics such as curvature, relative orientation to other features, and persistence in viewpoint space. For example, a T-junction can be described by a specific orientation and angle. Multiple features together give relative orientation constraints and also provide scale information. The contour topology itself can give a coarse estimate of the potential matching sets of viewpoints. For initial evaluation purposes, the implementation results presented in Section 5.3 make use of only T-junction orientation and angle information. Contour-contour interactions are selected from the rim appearance representation for the model based on precomputed orientation and angle geometry. Future work will add other features.

The angle formed by a T-junction between two edges is relatively stable over a small range of views (see Section 5.1). Since the CC-event is represented as a set of EE-events, the angle for a CC-event can be represented piecewise, where each edge pair has a fixed, precomputed angle. A CC-event and an image T-junction correspond where the measured angles of the two are similar. The set of CC-events is organized by this geometric measure so that portions of a CC-event with particular values can be selected efficiently.

In addition to the EE-event itself, a set of occluding contour fragments with high curvature values is stored. The orientation of each contour fragment is computed with respect to the coordinate system defined by the oriented EE-event. This small number of contour fragments provides an efficient template that can be used to verify the accuracy of a match between an EE-event and a detected T-junction. Precomputed contour fragments that are supported by evidence in the image increase the confidence of the match. Figure 5.6 shows an EE-event (circled) and the parts of the occluding contour that appear with high curvature values over the set of viewpoints where the EE-event persists. The relative geometry (with respect to the T-junction) of each of the occluding contour fragments is precomputed and stored with the EE-event.

In summary, the indexing phase of this algorithm selects candidate CC-events from the rim appearance representation based on the measured angle of a T-junction and the

Figure 5.6. The geometric relationship of high-curvature contour fragments to an EE-event reduces the number of CC-events that match a T-junction.

relative position and orientation of high-curvature portions of the contour with respect to the T-junction. The top candidates are passed on to the search component of the algorithm. Next, the search phase uses the candidate correspondences as starting points in order to compute a bounded region of viewpoints that best accounts for the observed image contours.

5.2.2. Refining CC-Event Correspondence

Once the initial correspondences are made between the model and an image feature, the persistence, geometric and co-occurring constraints described in Section 5.1.1 are applied to guide a local search for the boundaries of the solution region. Specifically, a candidate, qualitative correspondence between a CC-event and an image T-junction is refined by searching the EE-events that together define the CC-event. The EE-events, as members of the CC-event, may not satisfy the measured geometric properties of the image T-junction. Thus the hypothesized correspondence between a CC-event and a T-junction can be refined by searching the set of precomputed EE-events in the model through aspect space.

This search has been implemented as a fixed-distance search over the set of adjacent EE-events within the CC-event in aspect space. A fixed number of adjacent regions is examined to find a set of boundaries within the CC-event that best satisfies the

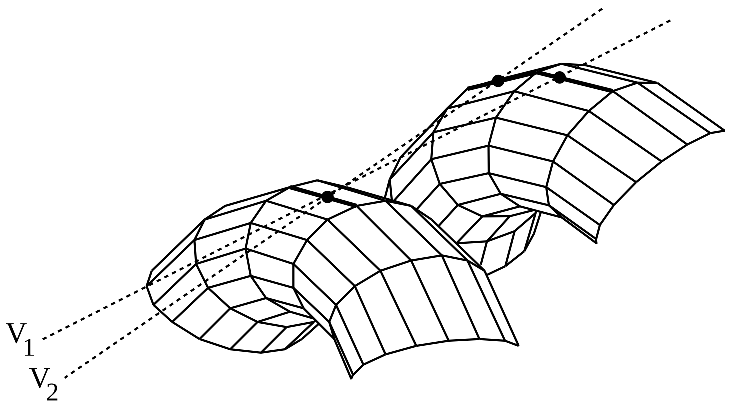


Figure 5.7. The two different EE-events that appear at $\mathbf{V}_1$ and $\mathbf{V}_2$ are connected together as part of the same CC-event. This is computed using edge connectivity and visual event adjacency information.

persistence, geometric and co-occurring constraints imposed by the image data and the hypothesized correspondence. The CC-event with the best measured match is selected as the solution region. Note that the search is over adjacent EE-events both spatially and in viewpoint. As an example of the latter case, Figure 5.7 shows two EE-events that lie on the same CC-event. These adjacencies in aspect space connect the EE-event at $\mathbf{V}_1$ to the EE-event at $\mathbf{V}_2$. Such viewpoint space adjacencies are also followed during the search for the boundaries of the solution region in viewpoint space.

5.3. Implementation Results

A prototype system has been implemented in order to study the issues involved with a model-based, explicit representation of occluding contour features. We have implemented the algorithms to compute the rim appearance representation under orthographic projection. The results reported here are twofold. First, the task of computing and storing the contour information for medium-sized polyhedral models is shown to be very manageable. Second, we describe a prototype implementation that recovers a viewpoint region estimate from a set of synthetically generated image features. All of the viewpoints in the solution viewpoint region produce an occluding contour that matches the image data.

Figure 3.12 shows the polyhedral model set. Table 5.1 indicates the model size, the total number of EE-events computed for each model, and the average persistence of each edge-edge interaction as a percentage of the total area of the view sphere. The number of edge interactions produced by a model is a function of several variables including the size of the model, the amount of non-convexity, and the sharpness of the angles between adjacent faces of the model.

There are two important observations from Table 5.1. First, the EE-event information is small enough to be efficiently computed and stored for a model with a moderate number of edges. The number of EE-events is O($\mathbf{n}^2$) where $\mathbf{n}$ is the number of edges. Rim constraints and non-degenerate polyhedral models give an average size complexity

Model	Edges	Computation Time (Secs)	Visible Patches	Average EE-Event Region Area
Chain link	144	4.8	2136	0.8
Small grid	105	1.7	490	1.7
Candlestick	560	32.3	4149	3.1
Candlestick and sphere	800	86.9	6734	2.0
Three spheres	720	108.3	2536	0.8
Interlocked tori	1024	129.8	6008	5.0
Torus and candlestick	1072	137.3	9335	2.9
Spring	2709	528	33029	2.3
Large grid	880	140.5	13486	1.6
Torus inside torus	1024	130.3	5494	4.4

Table 5.1. Size and time information for each of the polyhedral models in Figure 3.12. EE-event region area is the percentage of the entire area of the view sphere over which the EE-event occurs.

that is much smaller than this worst-case size. Second, the average persistence of EE-events is inversely proportional to the number of polyhedral faces. These observations illustrate the size/accuracy tradeoff that exists between large models that provide a very close approximation to a smooth surface and small models that provide a coarser approximation.

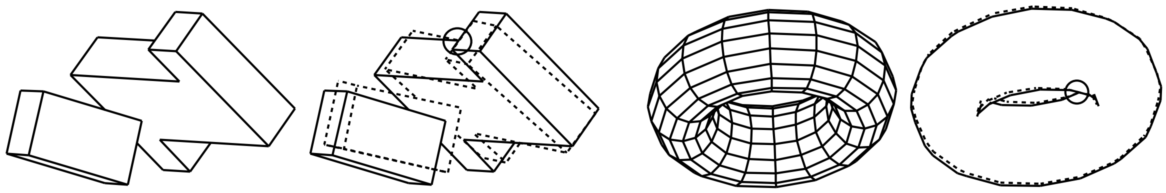
Figure 5.9 shows the computational results of determining a starting viewpoint given a synthetic image generated from an arbitrary viewpoint. The image contours were generated using orthographic projection with fixed (known) scale. Each image was represented as a set of individual edges, with known T-junctions. An approximate solution for a model-image match is a viewing direction (θ, ϕ) at the center of the solution EE-event region, a rotation α about the optical axis, and a translation τ in the image plane.

The circled T-junctions in Figure 5.9 were used as the primary constraint feature for correspondence with the precomputed representation of the occluding contour. The measured angle of the T-junction was used as the primary indexing key, with an initial match being any model T-junction with an angle within 10 degrees of the measured value. The relative position of the occluding contour to the oriented T-junction was used to further constrain potential correspondences. This is similar in style to the local feature focus method [Boll82]. A measure of the degree of correspondence was determined by using a least squares distance measure between predicted contour fragments and image contours.

An exact solution for viewpoint is not found, but rather a constrained region of viewpoints is found that accounts for the T-junction and occluding contour data. Given a tightly-constrained set of viewpoints, an exact solution can be found using, for example, an iterative method [Lowe87]. Furthermore, a contour correspondence is found, not an exact edge-to-edge correspondence. The two model edges aligned with the T-junction in Figure 5.9(d) are not the same model edges that originally projected to that T-junction. Because of the symmetry of the torus there are many close solutions that can be found. The solution viewpoint displayed is the center of the final solution viewpoint region. For our set of test models, all solution regions contained a true solution, and all viewpoints within each solution region were within $\pi/4$ radians of the exact solution.

Figure 5.9 shows the input image edges as solid lines, and the projected model at the solution viewpoint as dashed lines. The T-junction used as the central feature is circled. For the torus image, ten edge-edge correspondences within the CC-event for the circled T-junction were made by the algorithm before the solution shown in Figure 5.9(d) was found. The selected CC-event correspondence was revised after measuring the degree of match of the predicted occluding contour edges to the image. A correspondence was revised by using CC-event information in the model to obtain other edges that project to the same T-junction. This search was limited by the distance in the model from the original CC-event and the angle of the projected CC-event. Three CC-event correspondences were required to obtain the view shown of the S-shaped polyhedron in Figure 5.9(b).

An interesting problem occurs with "coarse" polyhedral models such as the S-shaped polyhedron shown in Figure 5.9(a). A coarse model is one that has large angles between adjacent faces so that it does not closely approximate a smooth surface. Sharp angles cause individual T-junctions to persist over larger ranges of viewpoint space and hence their projected angle values will each vary over a large range. As expected, large variations for a single EE-event makes any single choice in values for that EE-event very



(a) (b) (c) (d)

Figure 5.9. The solid lines in (a),(c) are the images used as input to the algorithm. The dashed lines in (b),(d) are the recovered views. The circled T-junctions in (b) and (d) were used as the initial feature for selecting a CC-event from the precomputed model of the occluding contour. The dashed lines are the projection of the model from the middle of the solution viewpoint region.

bad. For this case, an EE-event cannot be adequately represented by a single set of angle values. This again illustrates the time-space tradeoff in representational accuracy. Additional contour feature information incorporated into the algorithm can help to solve this problem.

We can make several observations and conclusions based on this prototype implementation. First, convex shapes provide no occlusion-based information, although the interaction between several convex polyhedra gives occlusion cues that constrain viewpoint. Hence, the curvature extrema and shape description of the occluding contour gives the only information about viewpoint. Second, symmetry generates a larger number of good model-image matches since symmetry generates a locus of viewpoints with identical contours. In this case, any of the possible correspondences on the locus of identical views is valid. Finally, the initial analysis of this approach has not incorporated other shape-based information that is almost always present in real 3D modeling situations. We have studied the occluding contour alone, with the future goal to add other shape information as well as texture, color and surface markings, to provide a larger set of constraints for viewpoint determination.

5.4. Concluding Remarks

Features of the occluding contour contain strong constraints for viewpoint determination. These constraints can be used to solve exactly for a region of viewpoints where a model will project to the observed occluding contour features. Our novel approach relies on the precomputed occluding contour features produced by polyhedra under orthographic projection. These precomputed features include contour T-junctions, contour terminals at concave edges and the relative arrangement of sections of the contour. The basis for the computation of this information is the construction, at all viewpoints, of the rim for polyhedra, defined as the analog of the rim for smooth shape. Precomputed contour information is organized over viewpoint based on edge connectivity in the model and feature adjacencies across viewpoint.

Occluding contour features are selected from the model given measured contour features and their relationships. Viewpoint constraints are associated with these qualitative correspondences since occluding contour features are defined by small regions of viewpoints where they can occur. The implementation results demonstrate that a model-based strategy using self-occlusion features such as T-junctions can efficiently constrain viewpoint by accounting for the occluding contour features detected in an image.

Future work will address issues related to integrating additional information for constraining viewpoint, including static features and feature changes with respect to viewpoint. A hybrid paradigm is currently being studied that integrates constraints from both occluding contour and occlusion features as well as surface discontinuity and surface-marking features. The problem of detecting occluding contour features and the sensitivity of this approach to noise and incomplete data is being studied by using dynamic information over time. The contour representation framework can incorporate constraints from the *changes* in contour with respect to viewpoint (i.e., time). We are also currently studying robust contour detection methods in image sequences and the model-based application of the derived contour dynamics.

This study of the rim appearance representation for model-based viewpoint determination has led to related research ideas that integrate rim appearance information with the interpretation tree paradigm. The notion of the interpretation tree with geometric constraints [Grim90] can be modified to include the specific geometric appearance information in the rim appearance representation. There has been work using the silhouette that is related to this idea [VanH87]. The following paragraphs briefly summarize the ideas relating the rim appearance to the interpretation tree paradigm for model-based vision.

An interpretation tree is an organized way of searching the possible correspondences between model and image features. A node in the tree corresponds to the pairing of a model feature (an edge, for example) to an image feature. A path from the root of the tree to a leaf is an interpretation, or an assignment of model parts to image features. Each level of the tree describes a partial interpretation of the image data. The central idea of the interpretation tree is that correspondences can be pruned by enforcing geometric constraints between assigned parts of the model [Grim90].

The primary power of the interpretation tree is derived from the strength of the constraints that prune the tree. The rim appearance representation provides a way to compute exact geometric constraints between parts of the occluding contour. The constraints between image fragments on the occluding contour are strong because the assumption that a fragment is on the occluding contour restricts the set of viewpoints to those where the model edges must lie on the rim. There are further constraints available in the relative geometry between parts of the occluding contour.

The rim appearance representation encodes geometric constraints for parts of the rim over the entire viewpoint space. We are working toward the efficient organization of the geometric constraints between parts of the rim for use with assigning consistent correspondences at each node in the interpretation tree. The two key elements that give this approach promise are (1) the efficient organization of geometric contraints in the rim appearance representation, and (2) the additional geometric constraints from features of self-occlusion, such as T-junctions.

Chapter 6

Summary

This thesis presents a novel, visual-event-based representation for 3D objects and a set of corresponding algorithms for solving problems in computer graphics and computer vision. Algorithms for the hidden-line and hidden-surface display of polyhedral scenes generated from a smooth path of viewpoints are developed to take advantage of viewpath coherence, the similarity in appearance between adjacent views of a scene. The rim appearance representation, a complete characterization of the structure of the occluding contour of polyhedra for all viewpoints, is introduced. An algorithm using the rim appearance representation has been implemented for the model-based vision problem of solving for a small set of viewpoints given the occluding contour of a 3D object. The following paragraphs give a chapter-by-chapter summary of the primary contributions of this thesis, along with a brief mention of current and future research directions.

Chapter 3 reviewed the asp representation [Plan88, Plan90], the basis for our display algorithms using visual events. The visual-event-based algorithm for display, first introduced by Plantinga [Plan88], has been developed, extended, implemented and tested. The problem of how to efficiently display a polyhedral scene over a path of viewpoints is cast as a problem of computing visual events along that path. A visual event is a viewpoint that causes a change in the structure of the image structure graph (ISG), a model's projected line drawing. Alternatively, a visual event is the set of viewpoints that cause a singularity in the projection map. The information stored with a visual event is sufficient to update a representation of the ISG. Thus the visible lines of a scene can be displayed as viewpoint changes by first precomputing and storing visual events, and then using those events at display time to interactively update the ISG.

Displaying visible surfaces requires the additional maintenance of ordering information with each visual event. This preserves the order of the visible edge segments bounding the visible part of each face. The added cost is a small constant times the number of faces in the scene. The benefit of the visible surface algorithm is the application of shaded-display techniques and shadows. The computation of shadow regions is a straightforward extension of the visible-surface algorithm since the appearance of the scene is precomputed for all viewpoints. Shadows are treated as polygons that are occluded from the light source's position.

The visible-line algorithm has been implemented for great circles on the view sphere under orthographic projection. The results on a set of eleven test models show that large models can be rotated in real time at frame rates greater than 20 frames per second. Only a small percentage of the total computation at display time is spent processing visual events, making the cost close to that of wire-frame display.

This method relies on a precomputation phase to compute visual events. There are three weaknesses to this approach that we are currently addressing. First, a large number of visual events can be generated in the worst case for pathological models such as grids. A hierarchically-structured model has been proposed to reduce the number of events by computing event data only for subparts of the model. Second, the representation is sensitive to changes, where a small change in the model can make it necessary to redo a large portion of the precomputation phase. The hierarchical model structure breaks the scene into components to limit the effect of small changes to the scene model. Finally, the extended algorithm using perspective projection computes visual events in a higher-dimensional space. We are working on a hybrid display algorithm that combines the depth-ordering display of faces with viewpath coherence. Only the EE-event is necessary in order to precompute those viewpoints where changes in occlusion relationships between faces occur. The EE-event boundaries can be computed and stored efficiently in $\mathbb{R}^3$ so that the potentially very high dimensionality of aspect space is avoided.

Chapter 4 introduced the rim appearance representation, a new, exact representation of the occluding contour for polyhedra. The geometry of self-occlusion for polyhedral edges is described, and an algorithm based on this geometry and on visual events is given for computing the visual event data for the occluding contour edges. While the worst-case time and space complexity for the rim appearance representation is the same as that of the asp, the average case results show a large decrease in the size of the representation. The details of this representation and its multi-level development in Chapter 5 is one of the major contributions of this thesis.

An algorithm for computing an approximation of the exact rim appearance representation is given. This approximate algorithm uses a simple global visibility test for EE-events, and ignores the EEE-event. By doing this the worst-case complexity in time and space is reduced from the exact algorithm at the cost of sacrificing exactness.

The algorithm for constructing the exact rim appearance representation has been implemented for orthographic projection. The sizes and times required for the construction of the rim appearance representation are much smaller than what is required for complete event data. The exact appearance of the occluding contour is preserved as a set of adjacent hypersurfaces in $\mathbb{R}^4$.

Chapter 5 describes an intermediate-level description of the behavior of the occluding contour over viewpoint. This description is based on the lower-level visual event data computed in the rim appearance representation. An algorithm for determining viewpoint given a set of occluding contours is described. This algorithm uses hypothesized correspondences between measurable features of the occluding contour as a way to formulate constraints on the possible views of the model. Three kinds of constraints from the EE-event include the persistence of the event, the geometry of the T-junction, and the geometry of pairs of features that co-occur.

The algorithm using the rim appearance representation for polyhedra has been implemented for the T-junction feature. The algorithm takes a projected image as input assuming that T-junctions and contours have been detected. The correspondence of an image T-junction with EE-events in the rim appearance representation gives a set of constraints in viewpoint space. These constraints lead to a solution that is a small set of viewpoints containing the original viewpoint.

One of the important contributions of this work is the development of the multi-level representation of the features of the occluding contour. This structure is in contrast to other viewer-centered approaches like the aspect graph that base the division of viewpoint space on global image properties such as the image structure graph (ISG) rather than individual features and their geometric relationships. We are interested in how to efficiently integrate the multi-level rim appearance representation with other model features like color, texture and surface markings, and with other known search paradigms. Specifically, the rim appearance representation can provide strong constraints for the interpretation tree paradigm [Grim90] for searching the space of model-image correspondences. The rim appearance representation makes occlusion and the appearance of the occluding contour explicit, while the interpretation tree efficiently organizes the search of correspondences and the application of constraints.

A second future direction of this work is a better understanding of the relationship between time and viewpoint. The change in image features over time can be detected with low-level representations such as spatiotemporal surfaces [Allm91]. The change in shape with respect to viewpoint that can be computed from a model (for example, the rim appearance representation) and its relationship to the change in shape that is observed in images over time is of great interest. This *spatiotemporal / spatio-viewpointal* problem is of interest because of the robustness of information over time and the added constraint that shape change provides for discrimination among 3D models.

References

[Allm91] Allmen, M., Toward spatiotemporal-motion-based recognition, Ph.D. Dissertation, Computer Science Department, University of Wisconsin-Madison, 1991.

[Athe78] Atherton, P., K. Weiler, and D. Greenberg, Polygon shadow generation, *Proc. SIGGRAPH*, 1978, 275-281.

[Bake88] Baker, H. H., Surface reconstruction from image sequences, *Proc. 2nd Int. Conf. Computer Vision*, 1988, 334-343.

[Barr81] Barrow, H. G. and J. M. Tenenbaum, Interpreting line drawings as three-dimensional surfaces, *Artificial Intell.* **17**, 1981, 75-116.

[Basr88] Basri, R. and S. Ullman, The alignment of objects with smooth surfaces, *Proc. 2nd Int. Conf. Computer Vision*, 1988, 482-488.

[Bish86] Bishop, G. and D. M. Weimer, Fast phong shading, *Proc. SIGGRAPH*, 1986, 103-106.

[Boll82] Bolles, R. and R. Cain, Recognizing and locating partially visible objects: The local-feature-focus method, *Int. J. Robotics Research* **1**(3), 1982, 57-82.

[Bowy89] Bowyer, K., D. Eggert, J. Stewman, and L. Stark, Developing the aspect graph representation for use in image understanding, *Proc. Image Understanding Workshop*, 1989, 831-849.

[Bowy91] Bowyer, K. W. and C. R. Dyer, Aspect graphs: an introduction and survey of recent results, *Int. J. Imaging Systems and Technologies*, to appear, 1991.

[Broo86] Brooks, F., Walkthrough – A dynamic graphics system for simulating virtual buildings, *Proc. Workshop on Interactive 3D Graphics*, 1986, 9-21.

[Burd85] Burde, G. and H. Zieschang, *Knots,* Walter de Gruyter, Berlin, 1985.

[Burn90] Burns, J. B., R. Weiss, and E. M. Riseman, View variation of point set and line segment features, *Proc. Image Understanding Workshop*, 1990, 650-659.

[Cipo90] Cipolla, R. and A. Blake, The dynamic analysis of apparent contours, *Proc. 3rd Int. Conf. Computer Vision*, 1990, 616-623.

[Cook84] Cook, R. L., T. Porter, and L. Carpenter, Distributed ray tracing, *Computer Graphics* **18**, July 1984, 137-145.

[Crow77] Crow, F. C., Shadow algorithms for computer graphics, *Proc. SIGGRAPH*, 1977, 242-248.

[Denb86] Denber, M. and P. Turner, A differential compiler for computer animation, *ACM Computer Graphics* **20**(4), 1986, 21-27.

[Dick91] Dickinson, S. J., A. P. Pentland, and A. Rosenfeld, From volumes to views: an approach to 3-D object recognition, *Proc. IEEE Workshop on Directions* *in Automated CAD-based Vision*, 1991, 85-96.

[Duff79] Duff, T., Smoothly shaded renderings of polyhedral objects on raster displays, *ACM Computer Graphics* **13**(2), 1979, 270-275.

[Egge89] Eggert, D. and K. Bowyer, Computing the orthographic projection aspect graph of solids of revolution, *Proc. IEEE Workshop on Interpretation of 3D Scenes*, 1989, 102-108.

[Farr85] Farrell, E., W. Yang, and R. Zappulla, Animated 3D CT imaging, *IEEE Computer Graphics and Applications* **5**(12), 1985, 26-32.

[Faug86] Faugeras, O. and M. Hebert, The representation, recognition, and locating of 3-D objects, *Int. J. Robotics Research* **5**(3), 1986, 27-52.

[Feke84] Fekete, G. and L. S. Davis, Property spheres: A new representation for 3-D object recognition, *Proc. IEEE Workshop on Computer Vision*, 1984, 192-201.

[Fole82] Foley, J. D. and A. VanDam, *Fundamentals of Interactive Computer Graphics,* Addison-Wesley, Reading, Mass., 1982.

[Fuch83] Fuchs, H., G. D. Abram, and E. D. Grant, Near real-time shaded display of rigid objects, *Computer Graphics* **17**, July 1983, 65-69.

[Gigu90] Gigus, Z. and J. Malik, Computing the aspect graph for line drawings of polyhedral objects, *IEEE Trans. Pattern Analysis and Machine Intell.* **12**(2), 1990, 113-122.

[Glas88] Glassner, A., Spacetime ray tracing for animation, *IEEE Computer Graphics and Applications* **8**(2), 1988, 60-70.

[Goad83] Goad, C., Special purpose automatic programming for 3-D model-based vision, *Proc. Image Understanding Workshop*, 1983, 94-104.

[Grim90] Grimson, W. E. L., *Object recognition by computer: The role of geometric constraints,* MIT Press, Cambridge, Mass., 1990.

[Grim90a] Grimson, W. E. L., The combinatorics of object recognition in cluttered environments using constrained search, *Artificial Intelligence* **44**, 1990, 121-165.

[Horn77] Horn, B. K. P., Understanding image intensities, *Artificial Intell.* **21**, 1977, 201-231.

[Hubs81] Hubschman, H. and S. W. Zucker, Frame-to-frame coherence and the hidden surface computation: Constraints for a convex world, *Computer Graphics* **15**, 1981, 45-54.

[Hutt90] Huttenlocher, D. P. and S. Ullman, Recognizing solid objects by alignment with an image, *Int. J. Computer Vision* **5**, 1990, 195-212.

[Ikeu87] Ikeuchi, K., Generating an interpretation tree from a CAD model for 3D object recognition in bin-picking tasks, *Int. J. Computer Vision*, 1987, 145-165.

[Joy88] Joy, K., C. Grant, N. Max, and L. Hatfield, *Tutorial: Computer Graphics: Image Synthesis*, IEEE Computer Society Press, Washington, D.C., 1988.

[Kass88] Kass, M., A. Witkin, and D. Terzopoulos, Snakes: active contour models, *Int. J. Computer Vision*, 1988, 321-331.

[Koen76] Koenderink, J. J. and A. J. van Doorn, The singularities of the visual mapping, *Biological Cybernetics* **24**, 1976, 51-59.

[Koen84] Koenderink, J. J., What does the occluding contour tell us about solid shape?, *Perception* **13**, 1984, 321-330.

[Koen87] Koenderink, J. J., An internal representation for solid shape based on the topological properties of the apparent contour, in *Image Understanding 1985-86*, W. Richards and S. Ullman, ed., Ablex, Norwood, N.J., 1987, 257-285.

[Koen90] Koenderink, J. J., *Solid Shape*, MIT Press, Cambridge, Mass., 1990.

[Korn87] Korn, M. R. and C. R. Dyer, 3-D multiview object representations for model-based object recognition, *Pattern Recognition* **20**, 1987, 91-103.

[Krie89] Kriegman, D. J. and J. Ponce, Computing exact aspect graphs of curved objects: Solids of revolution, *Proc. IEEE Workshop on Interpretation of 3D Scenes*, 1989, 116-122.

[Krie90] Kriegman, D. J. and J. Ponce, On recognizing and positioning curved 3D objects from image contours, *IEEE Trans. Pattern Analysis and Machine Intell.* **12**(12), 1990, 1127-1137.

[Kutu91] Kutulakos, K., *Personal communication*. 1991.

[Lowe87] Lowe, D. G., Three-dimensional object recognition from single two-dimensional images, *Artificial Intell.* **31**, 1987, 355-395.

[Lowe89] Lowe, D. G., Fitting parameterized 3-D models to images, Technical Report 89-26, Computer Science Department, University of British Columbia, Vancouver, B. C., 1989.

[Mali87] Malik, J., Interpreting line drawings of curved objects, *Int. J. Computer Vision* **1**, 1987, 73-103.

[Marr77] Marr, D., Analysis of occluding contour, *Proc. Royal Society London* **197**, 1977, 441-475.

[Murr89] Murray, D., D. Castelow, and B. Buxton, From image sequences to recognized moving polyhedral objects, *Int. J. Computer Vision* **3**, 1989, 181-208.

[Nalw88] Nalwa, V. S., Line-drawing interpretation: A mathematical framework, *Int. J. Computer Vision* **2**, 1988, 103-124.

[Nayl90] Naylor, B., SCULPT: An interactive solid modeling tool, *Proc. Graphics Interface '90*, 1990, 138-145.

[Newe72] Newell, M. E., R. G. Newell, and T. L. Sancha, A new approach to the shaded picture problem, *Proc. ACM Nat. Conf.*, 1972.

[Phon75] Phong, B. T., Illumination for computer generated pictures, *Comm. ACM* **18**, 1975, 311-317.

[Plan86] Plantinga, W. H. and C. R. Dyer, An algorithm for constructing the aspect graph, *Proc. IEEE Symp. Foundations of Computer Science*, 1986, 123-131.

[Plan87] Plantinga, W. H. and C. R. Dyer, The asp: A continuous viewer-centered representation for 3D object recognition, *Proc. 1st Int. Conf. Computer Vision*, 1987, 626-630.

[Plan88] Plantinga, W. H., The Asp: A Continuous, Viewer-centered Object Representation for Computer Vision, Ph.D. Dissertation, Computer Science Department, University of Wisconsin-Madison, 1988.

[Plan89] Plantinga, W. H., C. R. Dyer, and W. B. Seales, Real-time hidden-line elimination for a rotating polyhedral scene using the aspect representation, Technical Report 89-3, University of Pittsburgh, Pittsburgh, Pa., 1989.

[Plan90] Plantinga, W. H. and C. R. Dyer, Visibility, occlusion, and the aspect graph, *Int. J. Computer Vision* **5**(2), 1990, 137-160.

[Plan90a] Plantinga, W. H., W. B. Seales, and C. R. Dyer, Real-time hidden-line elimination for a rotating polyhedral scene using the aspect representation, *Proc. Graphics Interface '90*, 1990, 9-16.

[Ponc89] Ponce, J. and D. J. Kriegman, On recognizing and positioning curved 3D objects from image contours, *Proc. Image Understanding Workshop*, 1989, 461-470.

[Ponc90] Ponce, J. and D. J. Kriegman, Computing exact aspect graphs of curved objects: parametric surfaces, Technical Report UIUCDCS-R-90-1579, Department of Computer Science, University of Illinois at Urbana-Champaign, 1990.

[Ponc90a] Ponce, J. and D. J. Kriegman, Elimination theory and computer vision: recognition and the positioning of curved 3D objects from range, intensity, or contours, Technical Report UIUCDCS-R-90-1612, Department of Computer Science, University of Illinois at Urbana-Champaign, 1990.

[Rich88] Richards, W., B. Dawson, and D. Whittington, Encoding contour shape by curvature extrema, in *Natural Computation*, W. Richards, ed., MIT Press, Cambridge, Mass., 1988, 83-98.

[Rieg87] Rieger, J. H., On the classification of views of piecewise smooth objects, *Image and Vision Computing* **1**, 1987, 91-97.

[Seal90] Seales, W. B. and C. R. Dyer, Shaded rendering and shadow computation for polyhedral animation, *Proc. Graphics Interface '90*, 1990, 175-182.

[Shel82] Shelley, K. and D. Greenberg, Path specification and path coherence, *Computer Graphics* **16**(3), 1982, 157-166.

[Srip89] Sripradisvarakul, T. and R. Jain, Generating aspect graphs for curved objects, *Proc. Workshop on Interpretation of 3D Scenes*, 1989, 109-115.

[Stew88] Stewman, J. and K. Bowyer, Creating the perspective projection aspect graph of polyhedral objects, *Proc. 2nd Int. Conf. Computer Vision*, 1988, 494-500.

[Swan86] Swanson, R. W. and L. J. Thayer, A Fast-shaded polygon renderer, *Proc. SIGGRAPH*, 1986, 95-101.

[Thom87] Thompson, D. W. and J. L. Mundy, Three-dimensional model matching from an unconstrained viewpoint, *Proc. IEEE Int. Conf. Robotics and Automation*, 1987, 208-220.

[Thom85] Thompson, W. B., K. M. Mutch, and V. A. Berzins, Dynamic occlusion analysis in optical flow fields, *IEEE Trans. Pattern Analysis and Machine Intell.* **7**, 1985, 374-383.

[Toh90] Toh, P. S. and A. K. Forrest, Occlusion detection in early vision, *Proc. 3rd Int. Conf. Computer Vision*, 1990, 126-132.

[Vail89] Vaillant, R. and O. Faugeras, Using occluding contours for recovering shape properties of objects, *Proc. IEEE Workshop on Interpretation of 3D Scenes*, 1989, 26-32.

[VanH87] VanHove, P., Model-based silhouette recognition, *Proc. IEEE Workshop on Computer Vision*, 1987, 88-93.

[Wang90] Wang, R. and H. Freeman, Object recognition based on characteristic view classes, *Proc. 10th Int. Conf. Pattern Recognition*, 1990, 17-21.

[Whit55] Whitney, H., On the singularities of mappings in Euclidean spaces, I: mappings of the plane into the plane, *Ann. of Math.* **62**, 1955, 374-410.

[Whit80] Whitted, T., An improved illumination model for shaded display, *Comm. ACM* **23**, 1980, 343-349.

[Whit88] Whitted, T. and R. Cook, A comprehensive shading model, in *Computer Graphics: Image Synthesis*, K. Joy, C. Grant, N. Max and L. Hatfield, ed., IEEE Computer Society Press, Washington, D.C., 1988, 232-243.

[Worr89] Worrall, A. D., K. D. Baker, and G. D. Sullivan, Model based perspective inversion, *Image and Vision Computing* **7**(1), 1989, 17-23.

[Yan85] Yan, J., Advances in computer-generated imagery for flight simulation, *Computer Graphics and Applications* **5**(8), 1985, 37-51.

Abstract

A fundamental problem common to both computer graphics and model-based computer vision is how to efficiently model the appearance of a shape. Appearance is obtained procedurally by applying a projective transformation to a three-dimensional object-centered shape representation. This thesis presents a viewer-centered representation that is based on the visual event, a viewpoint where a specific change in the structure of the projected model occurs. We present and analyze the basis of this viewer-centered representation and the algorithms for its construction. Variations of this visual-event-based representation are applied to two specific problems: hidden line/surface display, and the solution for model pose given an image contour.

The problem of how to efficiently display a polyhedral scene over a path of viewpoints is cast as a problem of computing visual events along that path. A visual event is a viewpoint that causes a change in the structure of the image structure graph, a model's projected line drawing. The information stored with a visual event is sufficient to update a representation of the image structure graph. Thus the visible lines of a scene can be displayed as viewpoint changes by first precomputing and storing visual events, and then using those events at display time to interactively update the image structure graph. Display rates comparable to wire-frame display are achieved for large polyhedral models.

The rim appearance representation is a new, viewer-centered, exact representation of the occluding contour of polyhedra. We present an algorithm based on the geometry of polyhedral self-occlusion and on visual events for computing a representation of the exact appearance of occluding contour edges. The rim appearance representation, organized as a multi-level model of the occluding contour, is used to constrain the viewpoints of a three-dimensional model that can produce a set of detected occluding-contour features. Implementation results demonstrate that precomputed occluding-contour information efficiently and tightly constrains the pose of a model while consistently accounting for detected occluding-contour features.

Acknowledgments

First I would like to acknowledge the financial support this work has received from the University of Wisconsin Graduate School under Project Number 910288 and from the National Science Foundation under Grant Numbers IRI-8802436 and DCR-851228.

I am grateful to my advisor and mentor, Professor Chuck Dyer, for his guidance and support. He has sharpened my skills and has helped me progress as an academic. I thank the members of my thesis committee for the time and effort they spent reviewing this work. I am especially indebted to Dr. Kevin Bowyer for his willingness to travel from Florida to be a guest committee member. Thanks to my colleagues Harry Plantinga, Mark Allmen, Kyros Kutulakos, and David Luner, who have all been very helpful in discussing research ideas throughout my time in Madison, Wisconsin.

I have had a great deal of personal support from my friends in Madison and my family while I have been a graduate student. My involvement with InterVarsity, including friendships with Dr. Terry Morrison, Dr. Archie MacKinney and Cam Anderson has honed my insight and my outlook. Special thanks to the McKillops and the Schommers for our close ties, and to David Monsma for introducing me to cycling. I thank my parents, Jimmy and Janet Seales, for their gentle encouragement and wise foresight. Finally, I am profoundly grateful to my wife Pam, whose support for me extends far beyond my career.

Table of Contents

List of Figures

List of Tables

List of Equations