Finite Resolution Aspect Graphs of Polyhedral Objects

Ilan Shimshoni and Jean Ponce
Beckman Institute
University of Illinois
Urbana, IL 61801, USA

Abstract

We address the problem of computing the exact aspect graph of a polyhedral object observed by an orthographic camera with finite resolution, such that two image points separated by a distance smaller than a preset threshold cannot be resolved. Under this model, views that would be different under normal orthographic projection may become equivalent, while "accidental" views may occur over finite areas of the view space. We present a catalogue of visual events for polyhedral objects and give an algorithm for computing the aspect graph and enumerating all qualitatively different aspects. The algorithm has been fully implemented and results are presented.

1 Introduction

This paper addresses the problem of taking image resolution into account during the construction of the aspect graph of a polyhedral object [11]. Work in this area was pioneered by Kender and Freudenstein [9] and by Eggert et al.[5]. We focus on the case of an orthographic camera which cannot resolve image points closer than some preset distance and present a full implementation of an algorithm for computing the finite resolution aspect graph of a (not necessarily convex) three-dimensional polyhedron.

We believe that this is an important problem for two reasons. First, the size of aspect graphs increases very fast as a function of object complexity. For a polyhedron with n faces, the optimal data structure presented in [6] requires $O(n^6)$ space for storing the aspect graph. For an algebraic surface of degree d, the best bound on the size of the aspect graph established so far is $O(d^{12})$ [13]. The time complexity of the algorithms for constructing the aspect graph is at least as high [6, 7, 13, 14]. In practice, this means that one can only use aspect graphs of relatively simple objects. However by taking resolution into account we can simplify the aspect graph by disregarding features which cannot be seen due to the finite resolution of the camera.

Second, we believe that an even more severe problem is that aspect graphs fail to correctly predict the appearance of objects in actual images. In a sense, they are not complex enough: "accidental" views that are in theory unobservable are in fact routinely observed because the finite resolution of the camera blurs several image features into a single one [9].

We assume orthographic projection and suppose that two image points separated by a distance smaller than a preset threshold cannot be resolved. This is an appropriate model for a finite resolution camera observing objects from a roughly constant distance. It is much more limited than the full-perspective, multi-scale projection model used in [5], but it affords a practical algorithm for non-trivial three-dimensional polyhedral objects.

Informally, the aspect graph [11] is a qualitative, viewer-centered representation which enumerates all possible appearances of an object. More formally, choosing a camera model and a viewpoint determines the aspect of an object, i.e., the structure of the observed line-drawing.

The range of possible viewpoints can be partitioned into maximal connected sets (regions) that yield identical aspects. The change in aspect at the boundary between regions is called a visual event. The maximal regions and the associated aspects form the nodes of an aspect graph, whose arcs correspond to the visual event boundaries between adjacent regions.

Since their introduction by Koenderink and Van Doorn [11] more than ten years ago, aspect graphs have been the object of very active research. Most of it has focused on polyhedra: indeed, several algorithms have been proposed for computing the exact aspect graph of these objects under orthographic [2, 7, 14, 17] and perspective [19, 20, 21, 22] projection. Some of these algorithms have actually been implemented (e.g., [14, 17, 19, 20, 21, 22]). Recently, algorithms for constructing the exact aspect graph of curved objects such as solids bounded by quadric surfaces [3], solids of revolution [4], and algebraic surfaces [10, 13, 16, 18] have also been proposed and several of them have been implemented.

Previous approaches assume that the object of interest is observed with a camera having infinite resolution. Here, we address the case of a camera having finite resolution [5, 9]. We say that an edge is observable if its visible portion projects onto a segment of length greater than some fixed ϵ. Similarly two vertices are distinct when the distance between them in the image is greater than ϵ.

For infinite resolution cameras, the appearance of a polyhedral object changes at a set of visual events where two types of coincidental alignments occur [1, 7, 14, 20]: EV events, where a vertex projects onto the image of an edge, and EEE events, where the projections of three edges intersect at a single point.

By taking resolution into account, we will introduce new types of visual events related to EV and EEE events. They will correspond to the disappearance of an edge due to foreshortening alone, foreshortening and occlusion by a single edge (similar to an EV event), and to foreshortening due to occlusion by two edges (similar to a EEE event). At the same time, "accidental" views such as triple junctions of occluding edges will survive over finite areas of the view space. The next section presents a catalogue of these events.

2 Visual Events

We investigate what happens to the visual events of an infinite resolution aspect graph under the assumption of a finite resolution camera. Figure 1 shows the four visual events. The corresponding infinite resolution critical curve is transformed into a region of finite width whose boundaries form a new set of critical curves. We derive equations for these curves.

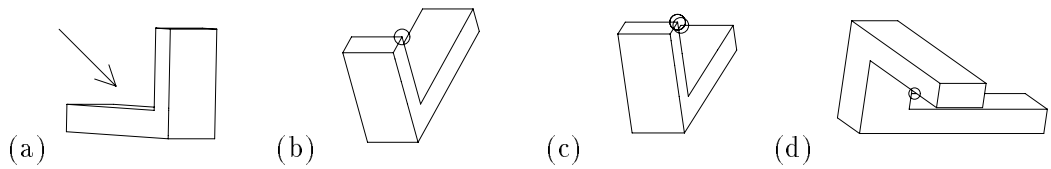

Figure 1: Four visual events: (a) a face begins to appear; (b) an EV event; (c) two vertices which are perceived as one vertex; (d) a EEE event.

2.1 Foreshortening of an Edge

An edge viewed from a direction parallel to it is seen as a point. Changing the viewing direction increases the edge's projected length, until it exceeds the camera resolution ϵ and the edge's

projection is seen as a line. The visual event occurs when the length is exactly ϵ. The same phenomenon happens when the viewing direction is parallel to the line connecting two nonadjacent vertices. We think of this line segment as an imaginary edge connecting the two vertices, therefore we treat this event in the same way as the foreshortening of an edge.

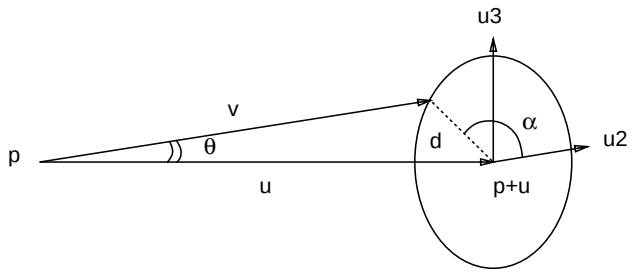

Figure 2: Illustration of the edge foreshortening event.

In both cases, when the distance between the projections of the vertices is less than ϵ, these appear to coincide in the image and the adjacent edges seem to emanate from the same point. Let v denote the viewing direction. The edge e is represented by the pair (p, u), where p is one extremity of the edge and $p+u$ is the other one. An edge is observable if and only if $|u \times v| \geq \epsilon |v|$.

The corresponding visual events correspond to views such that the angle between v and u is $\theta = \arcsin(\epsilon/|u|)$. They form opposite small circles on the viewing sphere, parameterized as follows: for each angle $0 \leq \alpha \leq 2\pi$ we find a point that when subtracted from the endpoint of the edge produces the desired viewing direction. As shown in Fig. 2, a parameterization of one of these circles is :

$$v(\alpha) = u + d(\cos\alpha u_2 + \sin\alpha u_3),$$

where u, u_2, u_3 is an orthonormal basis of $\mathbb{R}^3$ and $d = u\tan\theta$. The opposite circle is $-v(\alpha)$.

2.2 Edge-Vertex (EV) Event

With infinite resolution, an EV event occurs when the projection of a vertex lies on the projection of an edge. With a finite resolution camera, the vertex appears to be touching the edge when the distance from the projection of the vertex to the projection of the edge is less than ϵ (Fig. 3). There are two cases: if the vertex is in front of the face containing the edge, the infinite resolution event is replaced by two new events (Fig. 3(a-d)). If, on the other hand, the vertex is behind this face, we can not see the vertex when it is below the edge because it is occluded, therefore we will have one infinite resolution visual event plus one curve corresponding to the vertex being at a distance ϵ above the edge (Fig. 3(e-g)).

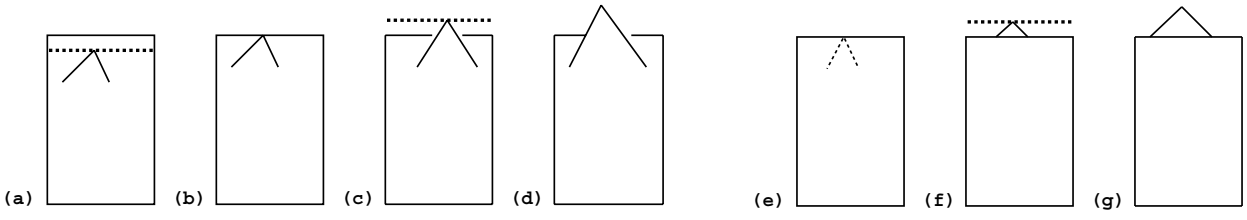

Figure 3: Finite resolution EV event. When the vertex is in front of the edge: (a) first finite resolution EV event; (b) infinite resolution EV event; (c) second finite resolution EV event; (d) vertex completely above the edge. When the edge is in front of the vertex: (e) infinite resolution EV event; (f) finite resolution EV event; (g) vertex completely above the edge.

Consider an edge e with endpoints e_1 and $e_2 = e_1 + u$ and a vertex p. A critical viewing direction is a vector v such that p is at distance ϵ from the plane that contains e_1 with normal $u \times v$. This condition can be written as:

$$(p - e_1) \cdot (u \times v) = \epsilon \mid u \times v \mid,$$

which is a quadratic equation in v.

We take a different approach for constructing a parametric representation. For every viewing direction where the distance from the projection of the vertex p to the projection of the edge e is ϵ, there must be a point on e whose projection is closest to the projection of p. That distance must be ϵ. We parameterize the edge e by $e(t) = (1-t)e_1 + te_2$, with $0 \leq t \leq 1$. For each $e(t)$, we compute a viewing direction $v(t)$ by finding a point $q(t)$ which satisfies the following conditions:

$$\begin{cases} |q(t) - e(t)| = \epsilon, & \\ (q(t) - e(t)) \perp u, & \text{since } e(t) \text{ is the closest point on } e \text{ to } q(t), \\ (q(t) - e(t)) \perp (q(t) - p), & \text{so that the distance in the image will be } \epsilon. \end{cases}$$

The first and third conditions characterize a circle on the sphere whose center is $e(t)$ and whose radius is ϵ. Intersecting that circle with the plane perpendicular to u which goes through $e(t)$ produces the two solutions of our problem. The above equation is adequate for finding viewing directions where $e(t)$ is internal to the edge. However, for the endpoint of the edge the second condition is relaxed. The result is a semicircle. This semicircle is the curve of the vertex coincidence event we discussed in the previous section.

It should also be noted that, for infinite resolution aspect graphs, EV events for which the vertex and the edge belong to the same face are normally not considered. Instead, special events corresponding to the appearance of a face when the viewing direction crosses the face's plane are considered separately. Here, this case is included in the EV events, and a face appears when the distance between the projection of at least one of its edges to one of its vertices becomes longer than ϵ (Fig. 4).

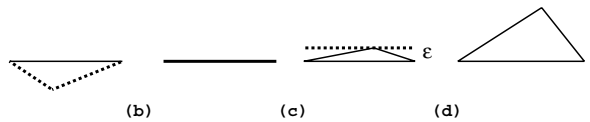

Figure 4: (a) The face is not visible. (b) Infinite resolution face-appearing event. (c) The finite resolution face-appearing event. (d) Face completely visible.

2.3 Edge-Edge-Edge (EEE) Event

Beside foreshortening, an edge may also seem to disappear because occlusion reduces its visible projected length to ϵ. This happens when two faces occlude the edge e and the distance between the intersection points of the corresponding edges with edge e is less than ϵ. In the image produced by a finite resolution camera all three edges appear to be intersecting at a point. This is the triple edge intersection event known as EEE (Fig. 5). We now characterize the views v such that the projection of the edges e_1 and e_2 intersect the projection of the edge e at a distance ϵ from each other.

To characterize intersections between straight lines in a simple way, we use Plücker coordinates, which describe a line by two orthogonal vectors (a, b) where a is some direction along the line, and if p is a point on the line, $b = p \times a$. We use the following property: two straight lines with Plücker coordinates (a_1, b_1) and (a_2, b_2) intersect if and only if $a_1 \cdot b_2 + a_2 \cdot b_1 = 0$.

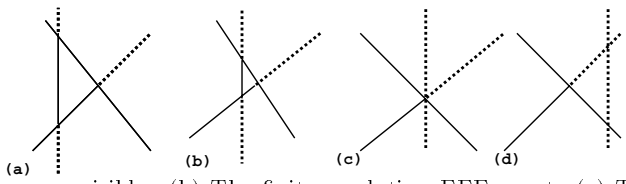

Figure 5: (a) The three edges are visible. (b) The finite resolution EEE event. (c) The infinite resolution EEE event. (d) Edge completely occluded.

We write that a line D_1' passing through $p + \lambda u$ with direction v intersects the supporting line $D_1 = (a_1, b_1)$ of e_1. We choose the origin in p, and the line D_1' has Plücker coordinates $(v, \lambda u \times v)$. Similarly, we write that a line D_2' passing through $p + (\lambda + \epsilon'/|u|)u$ with direction v intersects the supporting line $D_2 = (a_2, b_2)$ of e_2. The line D_2' has Plücker coordinates $(v, (\lambda + \epsilon'/|u|)u \times v)$. Writing that D_1' (resp. D_2') intersects D_1 (resp. D_2), we obtain two equations:

$$\begin{cases} \lambda a_1 \cdot (u \times v) + b_1 \cdot v = 0, \\ (\lambda + \epsilon'/|u|)a_2 \cdot (u \times v) + b_2 \cdot v = 0, \end{cases} \tag{1}$$

and eliminating λ among these yields:

$$\epsilon' = \frac{b_1 \cdot v|u|}{a_1 \cdot (u \times v)} - \frac{b_2 \cdot v|u|}{a_2 \cdot (u \times v)} = \epsilon \frac{|v||u|}{|u \times v|}. \tag{2}$$

while ϵ' is the actual distance between the two intersection points on e. Squaring this equation yields a homogeneous equation of degree 6 in v.

To generate points on the curve, we parameterize the edges e, e_1 and e_2 with s, t and u respectively. $e_1(t)$ and $e_2(u)$ are the intersection points between $e1$ and e on $e1$ and $e2$ and e on $e2$ respectively. $e(s)$ is the midpoint between these two intersection points on e. Given $e(s)$ we use an iterative method to compute $e_1(t)$ and $e_2(u)$. Given an initial estimate of ϵ' we recover $e_1(t)$,$e_2(u)$ and the viewing direction $v(s, t, u)$. From that we recompute ϵ'. This process usually converges in three to four iterations.

3 The Size of The Aspect Graph

It was shown in [14] that the complexity of the aspect graph for a polyhedral object of size n is $O(n^6)$. Using a similar argument we show that the size of the finite resolution aspect graph is also $O(n^6)$. There are $O(n)$ face curves, $O(n^2)$ vertex-vertex and edge-vertex curves and $O(n^3)$ edge-edge-edge curves. Because implicit equations of the curves are polynomials whose degree is bounded by a constant, a pair of curves will intersect at most a constant number of times. Therefore there are $O(n^6)$ vertices, edges, and regions in the finite resolution aspect graph. Because there are more finite resolution curves than infinite resolution curves and the degree of the implicit equations of the finite resolution curves is higher, the actual size of the finite resolution aspect graph will usually be larger but the asymptotic bound is still the same.

4 Aspect Graph Generation

We have discussed the different critical events and their equations, we will now show how to generate the aspect graph. The algorithm is divided into the following steps:

1. Generating the visual event curves from their equations.

2. Finding intersection points between curves.
3. Dealing with events which cannot be seen due to occlusion.
4. Constructing the stable regions and the corresponding aspects.

4.1 Generating the Visual Event Curves

As suggested in [6], we use a cube with edges of length 2 to represent the view space. Given the parametric representation of the visual event curve derived earlier, we construct a discrete representation of the visual event curve. We compute a viewing direction for discrete values of the parameter, then normalize the viewing directions such that $|v|_\infty = 1$, and trace them on the viewing cube.

4.2 Finding Intersections

We find intersection points between two curves using homotopy continuation [12]. In this case, the implicit equations of both curves are given to the algorithm. As we use the viewing cube we have to find the intersection points on the six faces of the sphere, therefore we run the algorithm six times, each time setting one of the coordinates of v to ± 1. The algorithm returns the intersection points. Points such that $|v|_\infty > 1$ are discarded. We compute the parameter value for the point and points with parameter value out of range are discarded.

4.3 Occluded events

Until now, a viewing direction has been considered to be part of a critical curve when the objects (edges, vertices) participating in the event satisfy the corresponding critical curve equation. However, in order for this viewing direction to be part of the actual critical curve, the participating objects must all be visible, or at least features adjacent to them must be visible. For example in Fig. 3(c) the position on the edge which is at distance ϵ away from the vertex is not visible but because we can see other parts of the edge, this viewing direction is part of the critical curve. A number of tests must be performed to check for occlusion.

First, the objects participating in the event can be occluded by a face lying in front of them. The point on the curve where the event becomes occluded is the point where an edge of another face starts to occlude the objects. In an EV event this point also lies on the EV curve of the same vertex and the edge of the occluding face. Therefore the place on the curve where occlusion starts is the intersection point between the two EV curves. In the EEE event a similar phenomenon happens and the place where the occlusion starts is the intersection point between two EEE curves.

Second, the objects participating in the event can also be occluded by faces adjacent to the objects themselves. In this case the place on the curve where the event becomes occluded is the point where the curve intersects one of the EV curves where the edge and the vertex belong to the occluding face.

4.4 Constructing the Regions and Aspects

We compute the regions from their bounding critical curves and curve intersections. We use a plane sweep algorithm [6] adapted for dealing with curves by adding critical events at extremal points, i.e., places where the curve is perpendicular to the sweeping direction. The algorithm computes the regions and returns a representative viewing direction for each region.

5 Simplifying The Aspect Graph

For each viewing direction we determine the finite resolution aspect and merge neighboring identical aspects. For each viewing direction we check which visual events are currently active (i.e. the distance between a set of features is at most ϵ). We attempt to merge close features to generate a simpler view of the object. When merging features we take into account that "nearness" is not transitive. For example in Fig. 6 when two edges are close to each other they appear as a thick edge in the image. As the distance between the edges is less than ϵ we can simplify the image of the thick edge to a thin edge. However in the second example edge v is close to edge u and to edge w but edges u and w are far from each other. Again our resulting image is a thick region but we can not simplify it to a line as in the previous example and we have to leave the black rectangle in the representation.

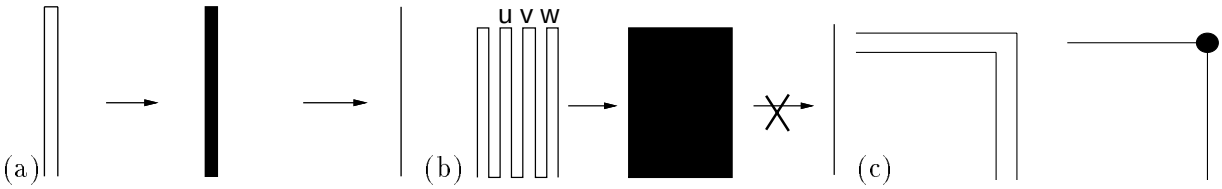

Figure 6: (a) Two close lines and their simplified image: (b) A comb shaped object and its simplified image: (c) A vertex which is close to two edges but not to the vertex which is adjacent to them and the simplified image with a "thick vertex".

In the initial stage of the algorithm, we build a graph representing the relationships between the edges and the vertices in the infinite resolution aspect, and a list of pairs of features that are near each other. There are various operations which can be done to incorporate pairs of features into the graph. For each such operation there are preconditions which must be fulfilled, changes to the graph, and the corresponding pairs of features which are removed from the list. For example if an edge is close to a vertex and transitivity of nearness is not violated the vertex is placed on the edge. Another example is shown in Fig. 6(c), in which a vertex is near two edges that are adjacent to a second vertex, which is not near the first vertex. Therefore transitivity of nearness is violated. Although the vertices are not close to each other we merge the edges and vertices. The resulting vertex is marked as a "thick vertex" because the image of that vertex will be thicker than a regular vertex. Operations have priorities assigned to them and the one with the highest priority is performed. The process is repeated until there are no more operations that can be performed.

In the last stage of the algorithm finite resolution aspects of neighboring regions are compared. Identical aspects are merged. The fact that adjacent regions would have identical aspects is a somewhat surprising phenomenon because we would expect different regions to have different aspects. Neighboring aspects turn out identical because different active events can have the same simplifying effect. For example in Fig. 7 an aspect in which a face has disappeared is produced from two neighboring aspects. These kinds of phenomena are very common and they reduce the size of the aspect graph considerably.

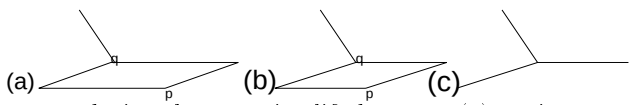

Figure 7: Two different aspects producing the same simplified aspect: (a) vertices p and q are near opposite edges; (b) vertices p and q are near each other and are near opposite edges; (c) The simplified aspect of both aspects.

6 Results

We present results obtained by running the algorithm on the object in Fig. 8(a). The infinite resolution visual event curves are shown in Fig. 8(b); there are 12 different regions in this case (compare to [6]). The visual event curves of the finite resolution aspect graph are shown in Fig. 8(c). These curves bound 377 different regions. The actual aspect graph is shown in Fig. 8(d): adjacent regions yielding the same aspect have been merged, and the resulting 117 regions are shown in different shades of grey.

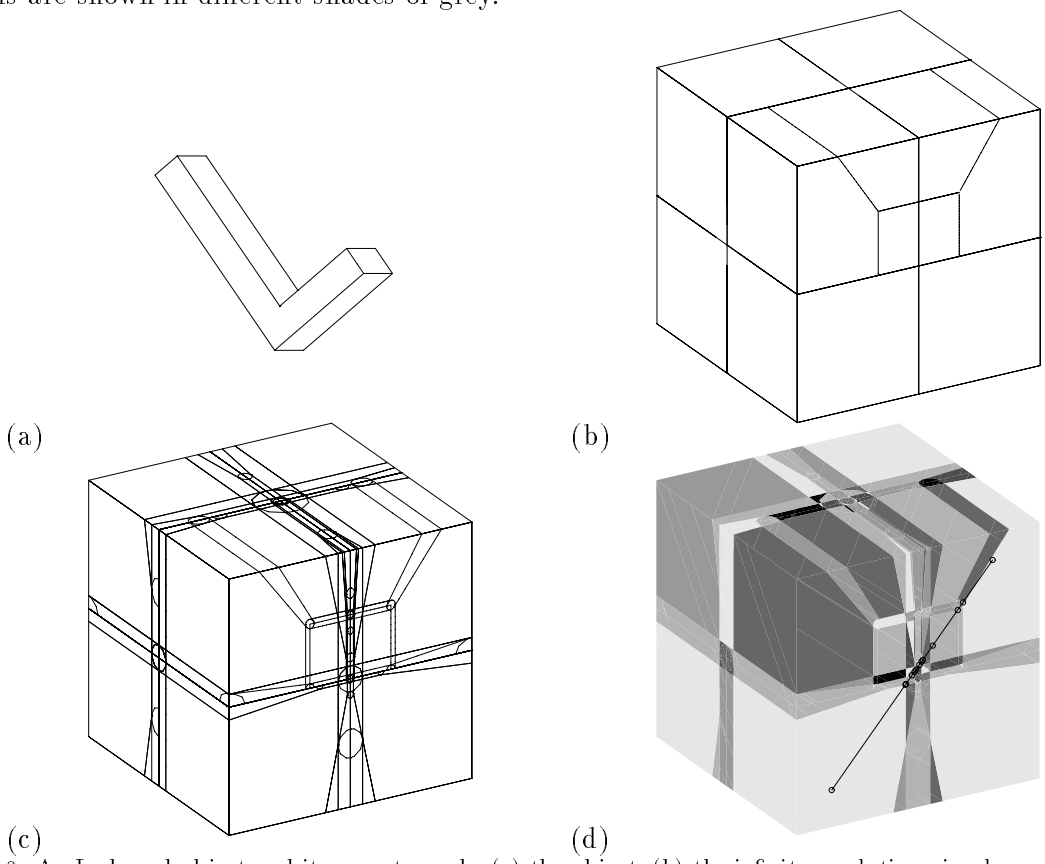

Figure 8: An L-shaped object and its aspect graph: (a) the object; (b) the infinite resolution visual event curves; (c) the finite resolution visual event curves; (d) the finite resolution aspect graph.

Figure 9 shows the qualitatively different aspects obtained by making a continuous sweep across some of the regions of the aspect graph. This sweep is indicated by the black diagonal line in Fig. 8(d), the viewing direction corresponding to each aspect being indicated by a small circle. The aspects in Fig. 9 have been rendered by merging features (vertices and/or edges) which are closer than ϵ. "Accidental" aspects now exist over a finite area of the viewing cube (e.g., Fig. 9.(i)). In other aspects (e.g., Fig. 9.(c,d)), the object seems to come from the Origami world [8]. Figure 10 shows the results obtained by running the algorithm on a more complex object which is shown in Fig. 10(a) and yields EEE curves. The visual event curves of the finite resolution aspect graph are shown in Fig. 10(b). The EEE curves are thicker. The aspect graph contains 357 regions, and it is shown in Fig. 10(c).

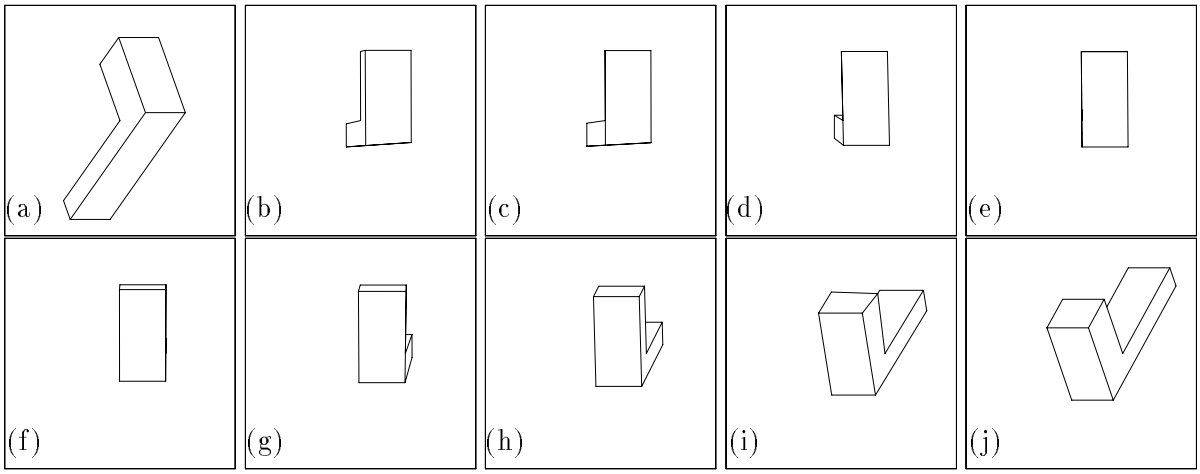

Figure 9: A series of aspects obtained by sweeping through adjacent views in the aspect graph.

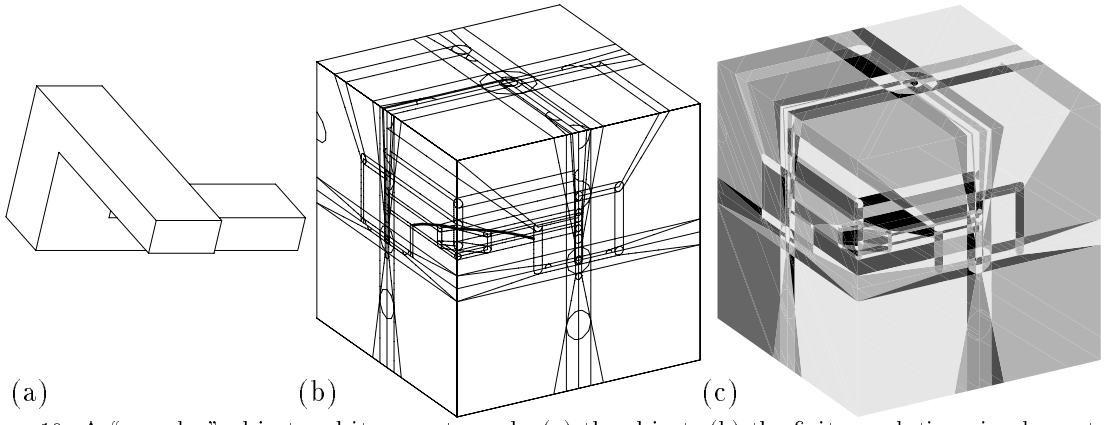

Figure 10: A "complex" object and its aspect graph. (a) the object: (b) the finite resolution visual event curves : (c) the finite resolution aspect graph.

7 Discussion

We have attacked the problem of computing the exact aspect graph of a polyhedral object observed by an orthographic camera with finite resolution, presented a catalogue of visual events for polyhedra observed under this projection model, and given an algorithm for computing the aspect graph and enumerating all qualitatively different aspects. Examples from the implementation of the algorithm have been presented.

One of the main problems with aspect graphs is that as the object gets more complex the aspect graph itself becomes unmanageably complex. At the resolution used in Fig. 8, this problem is certainly not solved by our approach, since the finite resolution aspect graph is larger than the infinite resolution one. As ϵ grows larger, or object features get smaller, some of the aspect graph features disappear: Figure 11 shows the finite resolution visual event curves of an object and the aspect graph for which some of the edges are shorter than ϵ. The aspect graph is considerably simpler than the previous one; in fact, some of the features appearing in the infinite resolution aspect graph have disappeared. The aspect graph which contains 74 regions is shown in Fig. 11(c). Comparing Figures 8(d) and 11(c); the aspect graph in Fig. 11(c) is definitely simpler. More work is required to compare the sizes of finite and infinite resolution aspect graphs as a function of resolution and object complexity.

Future work will be dedicated to actually using the finite resolution aspect graph in recognition tasks, maybe in conjunction with our work on qualitative line-drawing analysis in the

presence of uncertainty in vertex position [15]. The main challenge will be to better predict and interpret merged image features such as those appearing in Fig. 9.

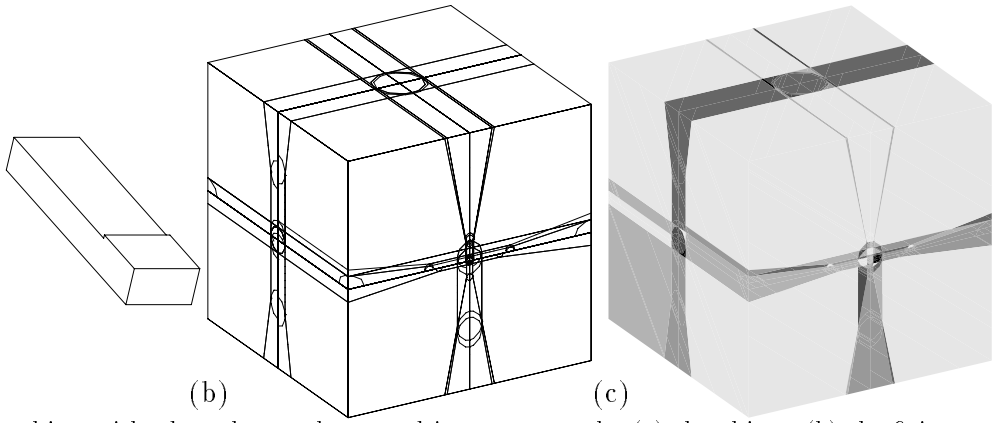

Figure 11: An object with edges shorter than ϵ and its aspect graph. (a) the object: (b) the finite resolution visual event curves : (c) the finite resolution aspect graph.

References

[1] K.W. Bowyer and C.R. Dyer. Aspect graphs: an introduction and survey of recent results. *International Journal of Imaging Systems and Technology*, 2:315–328, 1990.

[2] G. Castore. Solid modeling, aspect graphs, and robot vision. In Pickett and Boyse, editors, *Solid modeling by computer*, pages 277–292. Plenum Press, NY, 1984.

[3] S. Chen and H. Freeman. On the characteristic views of quadric-surfaced solids. In *IEEE Workshop on Directions in Automated CAD-Based Vision*, pages 34–43, June 1991.

[4] D. Eggert and K. Bowyer. Perspective projection aspect graphs of solids of revolution: An implementation. In *IEEE Workshop on Directions in Automated CAD-Based Vision*, pages 44–53, June 1991.

[5] D. Eggert, K. Bowyer, C. Dyer, H. Christensen, and D. Goldgof. The scale space aspect graph. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, pages 335–340, 1992.

[6] Z. Gigus, J. Canny, and R. Seidel. Efficiently computing and representing aspect graphs of polyhedral objects. *IEEE Trans. Patt. Anal. Mach. Intell.*, 13(6), June 1991.

[7] Z. Gigus and J. Malik. Computing the aspect graph for line drawings of polyhedral objects. *IEEE Trans. Patt. Anal. Mach. Intell.*, 12(2):113–122, February 1990.

[8] T. Kanade. A theory of the Origami world. *Artificial Intelligence*, 13:279–311, 1980.

[9] J. Kender and D. Freudenstein. What is a degenerate view. In *Proc. International Joint Conference on Artificial Intelligence*, pages 801–804, August 1987.

[10] Y.L. Kergosien. Generic sign systems in medical imaging. *IEEE Computer Graphics and Applications*, 11(5):46–65, 1991.

[11] J.J. Koenderink and A.J. Van Doorn. The internal representation of solid shape with respect to vision. *Biological Cybernetics*, 32:211–216, 1979.

[12] A.P. Morgan. *Solving Polynomial Systems using Continuation for Engineering and Scientific Problems.* Prentice Hall, Englewood Cliffs, NJ, 1987.

[13] S. Petitjean, J. Ponce, and D.J. Kriegman. Computing exact aspect graphs of curved objects: Algebraic surfaces. *Int. J. of Comp. Vision*, 9(3), 1992.

[14] H. Plantinga and C. Dyer. Visibility, occlusion, and the aspect graph. *Int. J. of Comp. Vision*, 5(2):137–160, 1990.

[15] J. Ponce and I. Shimshoni. An algebraic approach to line-drawing analysis in the presence of uncertainty. In *IEEE Int. Conf. on Robotics and Automation*, pages 1786–1791, 1992.

[16] J.H. Rieger. Global bifurcations sets and stable projections of non-singular algebraic surfaces. *Int. J. of Comp. Vision*, 7(3):171–194, 1992.

[17] W.B. Seales and C.R. Dyer. Constrained viewpoint from occluding contour. In *IEEE Workshop on Directions in Automated "CAD-Based" Vision*, pages 54–63, Maui, Hawaii, June 1991.

[18] T. Sripradisvarakul and R. Jain. Generating aspect graphs of curved objects. In *Proc. IEEE Workshop on Interpretation of 3D Scenes*, pages 109–115, Austin, TX, December 1989.

[19] J. Stewman and K.W. Bowyer. Aspect graphs for planar-face convex objects. In *Proc. IEEE Workshop on Computer Vision*, pages 123–130, Miami, FL, 1987.

[20] J. Stewman and K.W. Bowyer. Creating the perspective projection aspect graph of polyhedral objects. In *Proc. Int. Conf. Comp. Vision*, pages 495–500, Tampa, FL, 1988.

[21] R. Wang and H. Freeman. Object recognition based on characteristic views. In *International Conference on Pattern Recognition*, pages 8–12, Atlantic City, NJ, June 1990.

[22] N. Watts. Calculating the principal views of a polyhedron. CS Tech. Report 234, Rochester University, 1987.