[20] I. Shimshoni and J. Ponce. Recovering the shape of polyhedra using line-drawing analysis and complex models. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, pages 514–519, Seattle, Washington, June 1994.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

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

[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. *Géométrie énumérative et contacts de variétés linéaires: application aux graphes d'aspects d'objects courbes*. PhD thesis, Institut National Polytechnique de Lorraine, 1995.

[14] 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.

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

[16] F. P. Preparata and M.I. Shamos. *Computational Geometry, An Introduction*. Springer-Verlag, New York, 1985.

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

[18] J.H. Rieger. On the complexity and computation of view graphs of piecewise-smooth algebraic surfaces. Technical Report FBI-HH-M-228/93, Universität Hamburg, 1993.

[19] 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.

[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. W. Eggert, K. W. Bowyer, C. R. Dyer, H. I. Christensen, and D. B. Goldgof. The scale space aspect graph. *IEEE Trans. Patt. Anal. Mach. Intell.*, 15(11):1114–1139, 1993.

[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] D. Kahaner, C. Moler, and S. Nash. *Numerical Methods and Software*. Prentice Hall, Englewood Cliffs, NJ, 1989.

[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.

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 [20]. The main challenge will be to better predict and interpret merged image features such as those appearing in Figure 18.

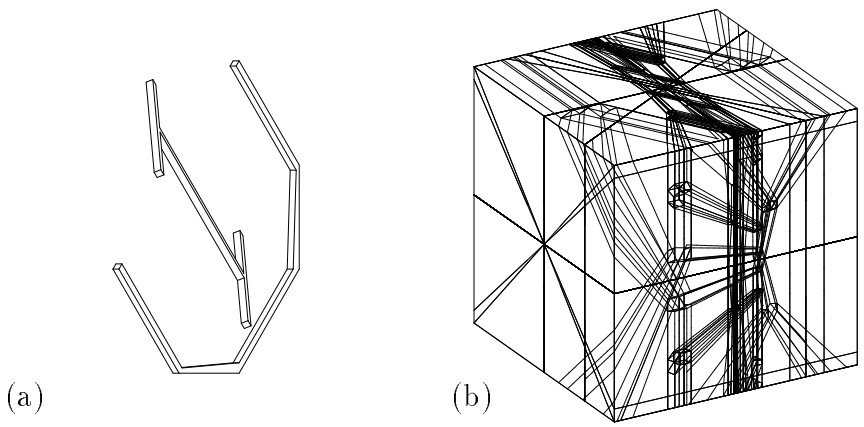

Figure 23: A very non-convex object and its infinite-resolution visual event curves: (a) the object; (b) the infinite-resolution visual event curves.

Acknowledgements: This work was supported in part by the Beckman Institute and the Center for Advanced Study of the University of Illinois at Urbana-Champaign, by the National Science Foundation under grant IRI-9224815, and by the National Aeronautics and Space Administration under grant NAG 1-613.

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.

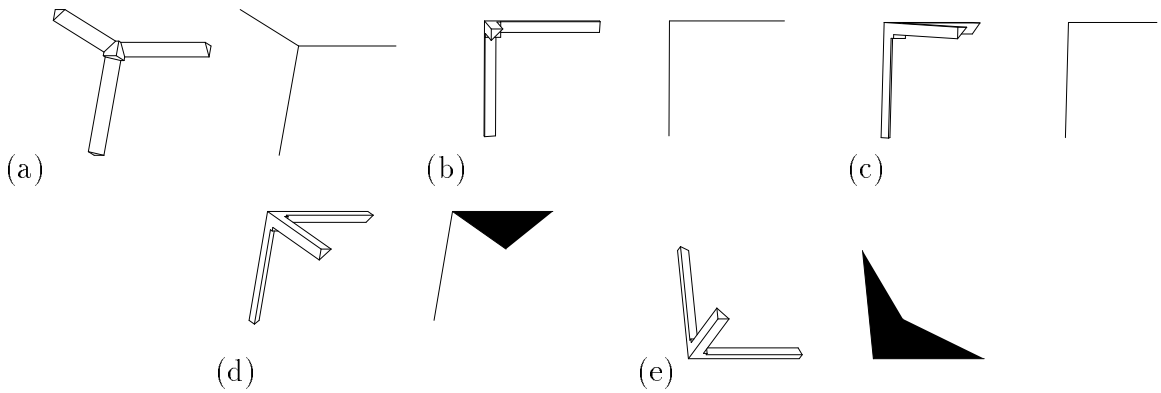

Figure 22: All qualitatively different finite-resolution aspects of an object consisting of three orthogonal legs: (a) all three legs are visible in the finite-resolution aspect; (b) one of the legs is foreshortened to a point in the finite-resolution aspect; (c) two legs appear as one line in the finite-resolution aspect; (d) two legs are close to each other but the width of the combined feature is more than ϵ and therefore they appear as a region in the finite-resolution aspect; (e) one of the legs is close to the other two legs but they are not close to each other causing all three of them to appear as a region in the finite-resolution aspect.

the aspect graph itself becomes unmanageably complex. An example of such an object and its infinite-resolution visual event curves is shown in Figure 23. The more features in the object that are visible from other features, the more complex the aspect graph becomes. The problem is even more acute when trying to generate finite-resolution event curves. In this example the cost was so prohibitive that we were not able to generate the finite-resolution visual event curves. So even though the final finite-resolution aspect graph can be quite simple as was shown in the example in Figure 21, the cost of producing the finite-resolution aspect graph could be prohibitively high. Thus more work is required to find ways to prune the number of finite-resolution event curves to enable us to efficiently compute finite-resolution aspect graphs of more complex objects. More work is also required to compare the sizes of finite- and infinite-resolution aspect graphs as a function

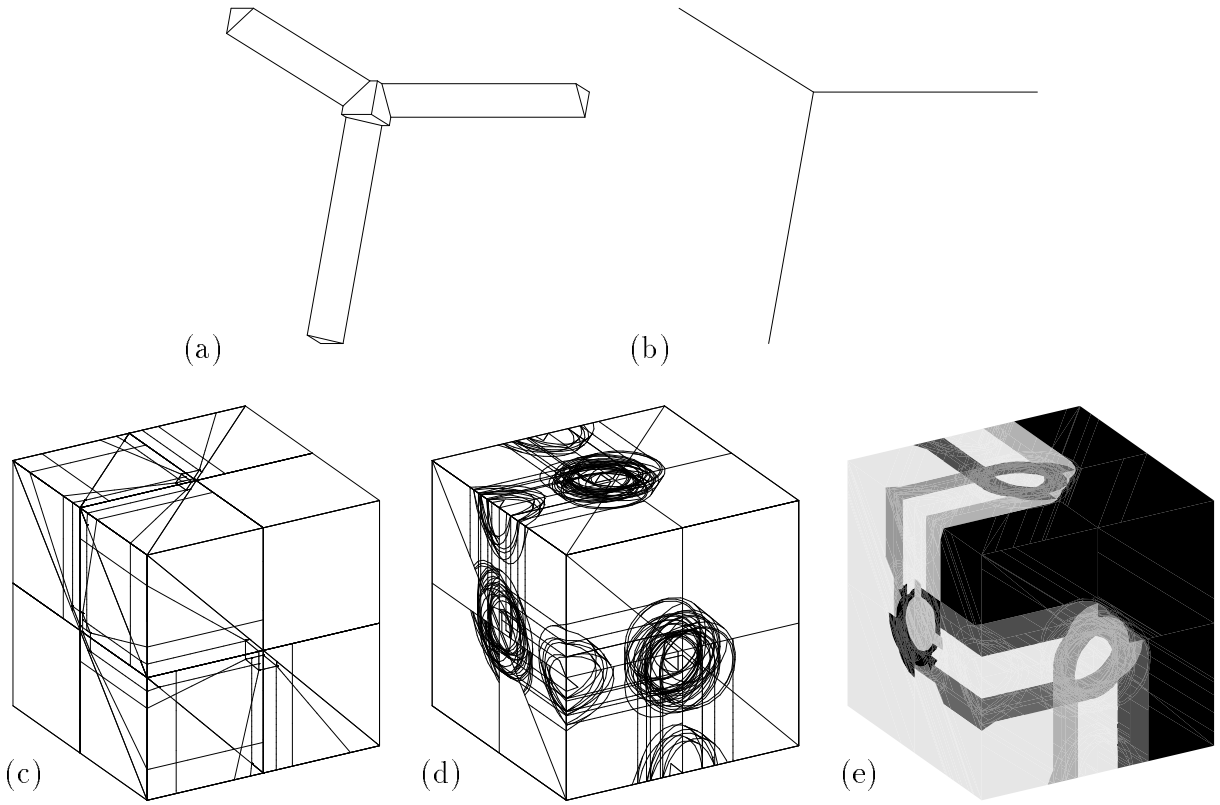

Figure 21: An object consisting of three orthogonal legs and its aspect graph: (a) the object; (b) its finite-resolution aspect; (c) the infinite-resolution visual event curves; (d) the finite-resolution visual event curves; (e) the finite-resolution aspect graph.

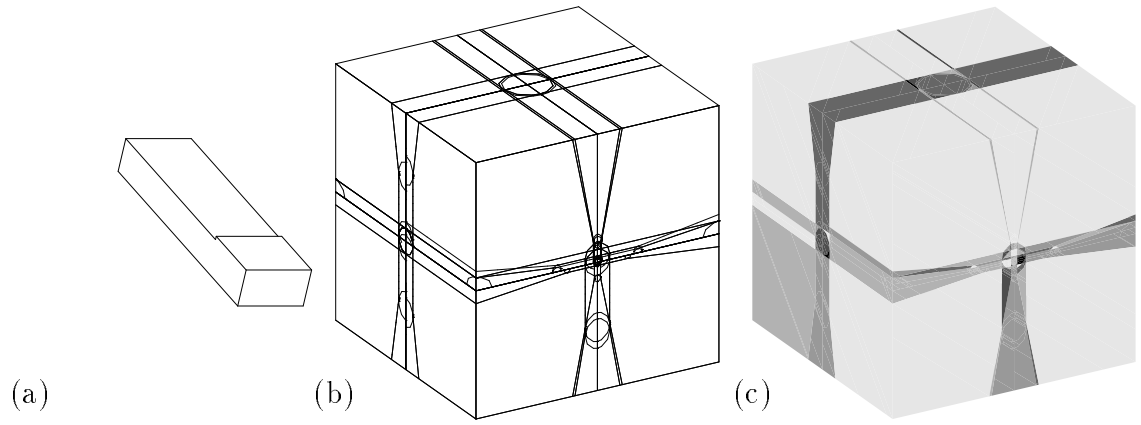

Figure 20: An object with edges shorter than ϵ and its aspect graph: (a) the object; (b) the finite-resolution visual event curves; (c) the finite-resolution aspect graph.

the number of regions is reduced to 20 (Figure 21(e)), which is considerably less then for the infinite-resolution aspect graph. The finite-resolution aspect graph is smaller than the infinite-resolution aspect graph in this case because the infinite-resolution aspect graph has to account for the different appearances of each leg and other small changes in appearance which do not exist in the finite-resolution aspects. A catalogue of all qualitatively different finite-resolution aspects is shown in Figure 22.

8 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

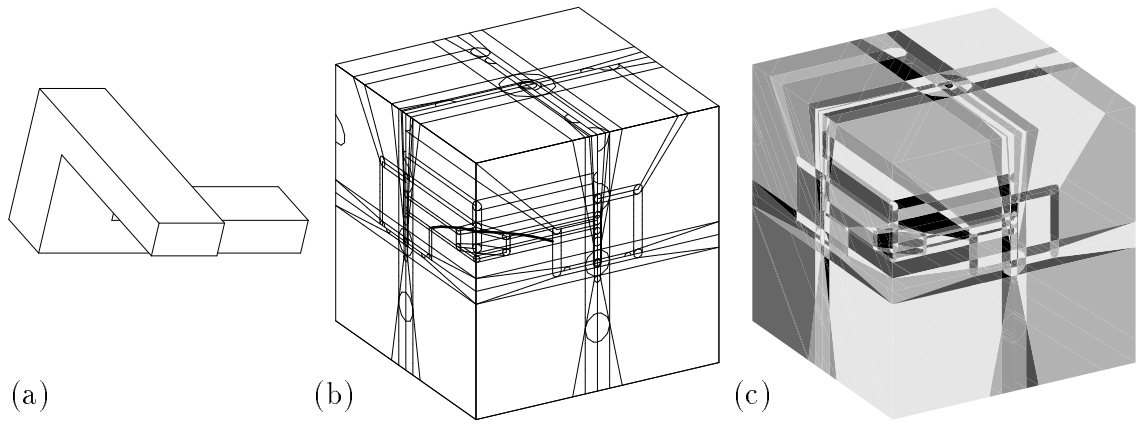

Figure 19: A "complex" object and its aspect graph: (a) the object; (b) the finite-resolution visual event curves; (c) the finite-resolution aspect graph.

the aspect graph itself becomes unmanageably complex. At the resolution used in Figure 17, 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 20 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 more simple than the previous one; in fact, the features due to the short edges appearing in the infinite-resolution aspect graph have disappeared. The aspect graph (which contains 74 regions) is shown in Figure 20(c). The aspect graph in Figure 20(c) is definitely simpler than the one shown in Figure 17(d).

A more dramatic effect of the simplification power of the algorithm is shown in Figure 21. The object shown in Figure 21(a) consists of three connected orthogonal "legs". At the resolution at which we computed the finite-resolution aspect graph, the legs simplify to lines (Figure 21(b)). The infinite-resolution aspect graph shown in Figure 21(c) contains 120 regions. The finite-resolution visual event curves shown in Figure 21(d) bound 5060 regions, however after merging regions which have the same finite-resolution aspect,

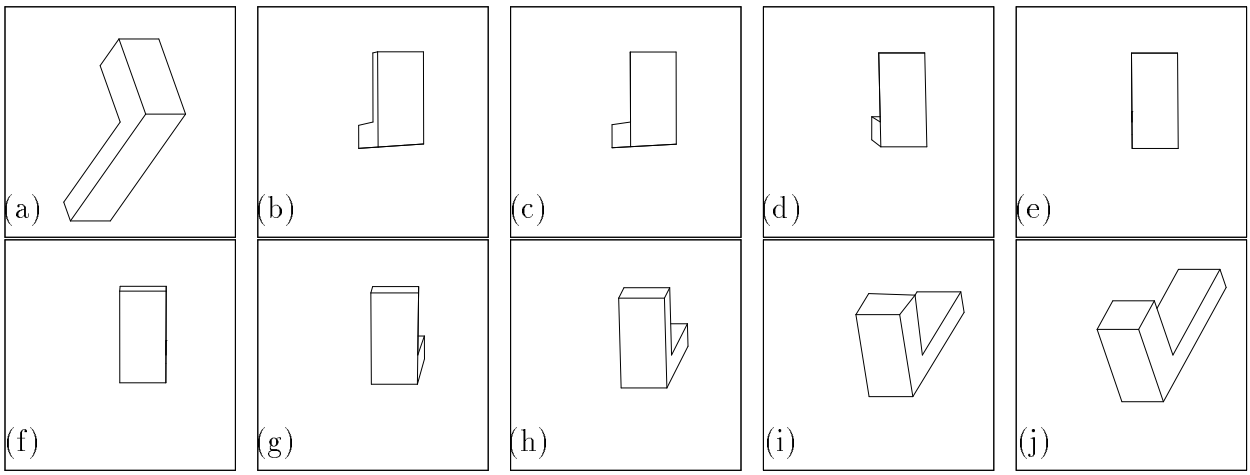

Figure 18: A series of aspects obtained by sweeping through adjacent views in the aspect graph.

different shades of grey.

Figure 18 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 Figure 17(d). The viewing direction corresponding to each aspect is indicated by a small circle. The aspects in Figure 18 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., Figure 18.(i)). In other aspects (e.g., Figure 18.(c,d)), the image does not look like a real aspect of the object because the finite-resolution aspect is intended to convey relationships between edges and vertices and not the image of the object itself.

Figure 19 shows the results obtained by running the algorithm on a more complex object (Figure 19(a)) which yields EEE curves. The visual event curves of the finite-resolution aspect graph are shown in Figure 19(b). The EEE curves are shown as heavy lines. The aspect graph shown Figure 19(c) contains 357 regions.

One of the main problems with aspect graphs is that as the object gets more complex

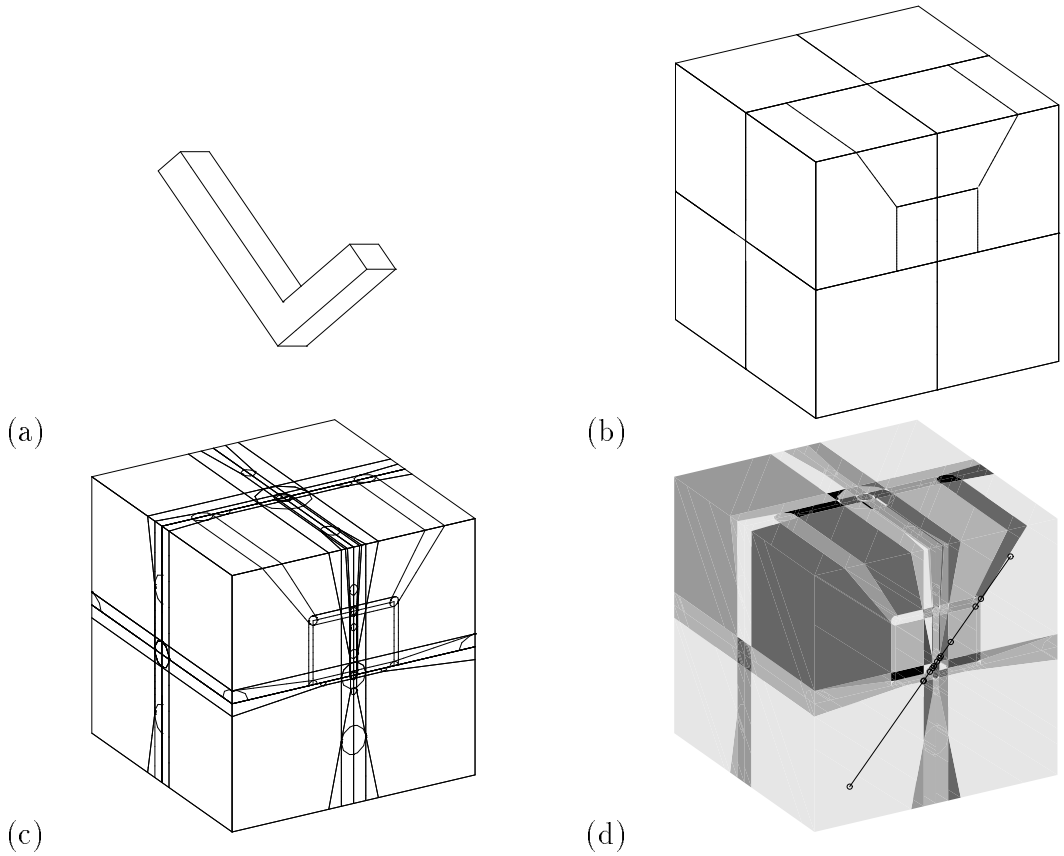

Figure 17: An L-shaped object and its aspect graph: (a) the object; (b) the infinite-resolution visual event curves; (c) the finite-resolution visual event curves; (d) the finite-resolution aspect graph.

7 Results

In this section we present results obtained by running the algorithm on several objects. the infinite-resolution visual event curves or the object shown in Figure 17(a) are shown in Figure 17(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 Figure 17(c). These curves bound 377 different regions. The aspect graph is shown in Figure 17(d): adjacent regions yielding the same aspect have been merged, and the resulting 117 regions are shown in

$|F|$ is the number of edges vertices and regions respectively. 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.

We use an argument similar to the one in [6] to analyze the complexity of the algorithm for computing the finite-resolution aspect graph. The time required to generate the $O(n^3)$ curves is $O(n^3)$. For each curve we determine which segments of the curve represent views in which the event is visible by finding the points on the curve where the features participating in this event become occluded by an edge. Therefore each of the $O(n^3)$ curves is intersected with $|E| = O(n)$ curves yielding $O(n^4)$ intersection points at a cost of $O(n^4)$. Sorting the intersection points and determining the actual segments of visible event curves has the complexity of $O(n^4 \log n)$. As described above the number of vertices, edges and regions in the aspect graph is $O(n^6)$, however most aspect graphs are much smaller. For an aspect graph of actual size m, the time for running the plane-sweep algorithm for constructing it is $O(m \log m)$.

The overall time complexity for tracing the curves and finding the regions of the aspect graph is $O(n^4 \log n + m \log m)$. The complexity of generating the aspect graph is the same as for the algorithm to generate the infinite-resolution aspect graph described in [6]. However our algorithm has an additional simplification stage which has to be run on a representative of each of the m regions of the finite-resolution aspect graph. In that stage the object must be backprojected and "active" events must be found and processed. This stage whose time complexity is at least $O(m\ n)$ is larger than the complexity of the previous stages and dominates the complexity of algorithm.

two close vertices are merged and the corresponding edge is removed. In Figure 16(b) a very narrow face is removed. Although a number of vertices and edges are close to each other the more complex operation of face removal is performed. Figure 16(d) is a more complicated example of the same operation. In Figure 16(c), two sets of close edges are merged into only two edges, and in Figure 16(e) an example of the "thick vertex" and two face removal operations is shown.

In the last stage of the algorithm, finite-resolution aspects of neighboring regions are compared. Identical aspects are merged. In order to merge two regions the two graphs have to be isomorphic and the vertices from the infinite-resolution aspect which appear in the finite-resolution aspect have to map to the same vertices in the finite-resolution aspects.

6 Size of The Aspect Graph and Algorithm Time Complexity

It was shown in [15] that the complexity of the aspect graph for a polyhedral object of size n is $O(n^6)$. We use a similar argument to show that the size of the finite-resolution aspect graph is also $O(n^6)$. There are $O(n^2)$ vertex-vertex and edge-vertex curves and $O(n^3)$ edge-edge-edge curves. By Bezout's theorem the number of intersection points between several polynomials is the product of their degrees; since the degree of the implicit equations of the curves is bounded by a constant (i.e., 6), a pair of curves will intersect on the viewing sphere at most a constant (i.e., 72) number of times. Therefore each curve will be intersected by other curves in $O(n^3)$ places yielding $O(n^3)$ edges and vertices. For all curves there will be $O(n^6)$ edges and vertices. The number of regions which is also $O(n^6)$ is found using Euler's formula $|F| = 2 - |V| + |E|$, where $|E|, |V|$ and

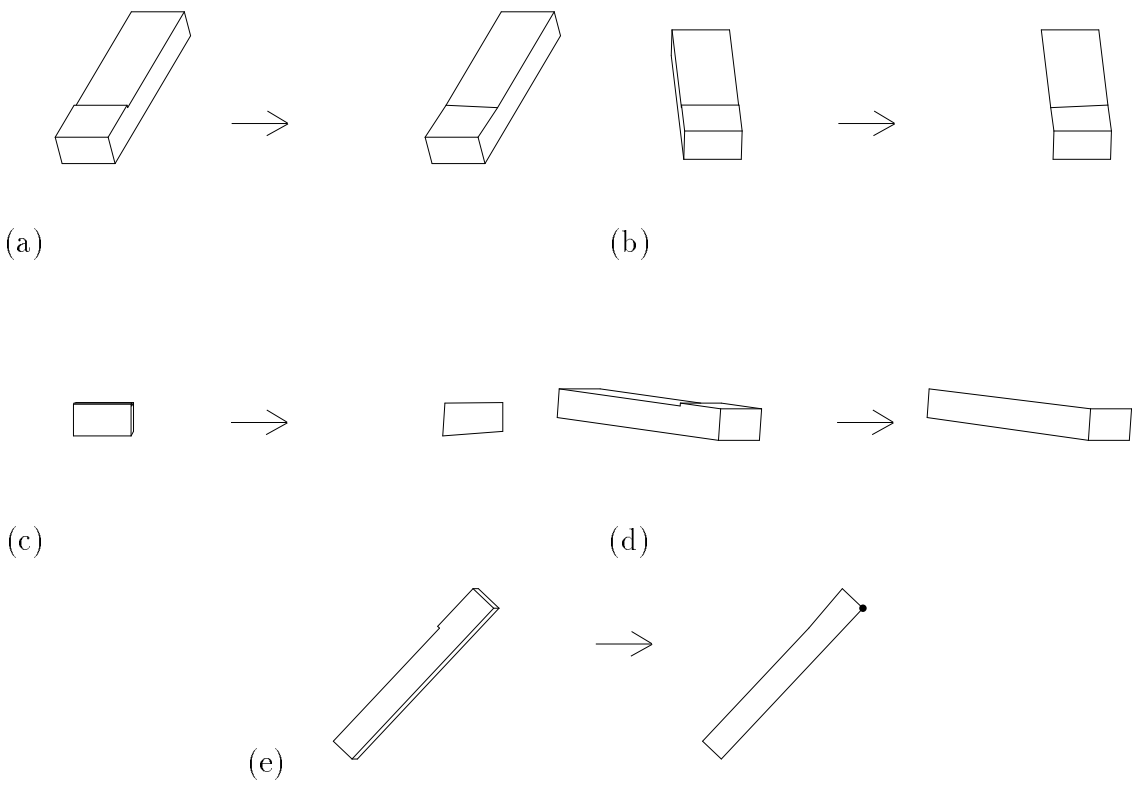

Figure 16: Examples of infinite-resolution aspects and their simplified aspects: (a) a short edge removed; (b) a narrow face is removed; (c) many close edges merged; (d) narrow faces removed; (e) a "thick" vertex created.

For each such operation there are preconditions which must be fulfilled; changes are made to the graph, and the corresponding pairs of features are removed from the list. Complex operations were defined only when their effect could not be achieved by applying a series of simpler operations. Operations are sorted by complexity, and the most complex operation whose preconditions have been met is performed. The process is repeated until there are no more operations that can be performed.

In Figure 16 a number of examples of aspect simplification are shown. In Figure 16(a)

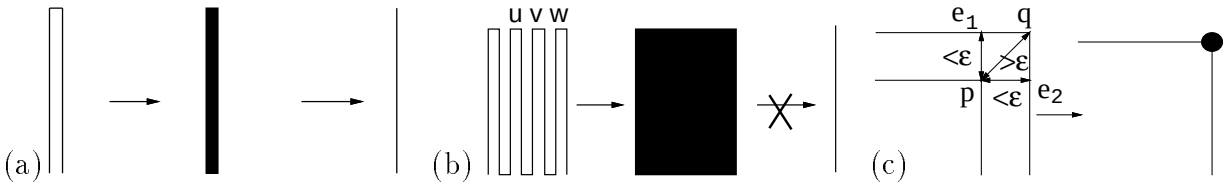

Figure 14: Examples of aspect simplification: (a) two close lines and their simplified image: (b) a comb-shaped object and its simplified image: (c) a vertex which is close to two edges but not to the vertex which is adjacent to them and the simplified image with a "thick vertex".

boring aspects may turn out identical because different active events can have the same simplifying effect. For example in Figure 15 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.

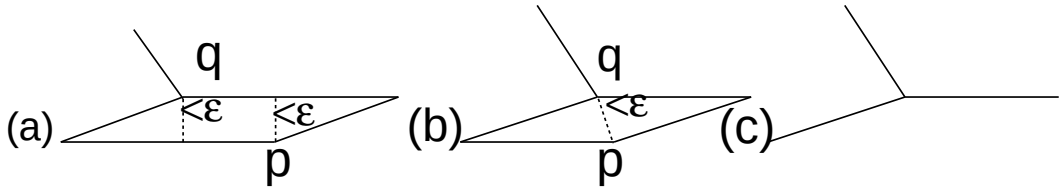

Figure 15: Two different aspects producing the same simplified aspect: (a) vertices p and q are near opposite edges; (b) vertices p and q are near each other and are near opposite edges; (c) The simplified aspect.

5.2 Aspect Simplification

In the initial stage of the algorithm, we build a graph representing incidence relationships between the edges and 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. They can be as simple as merging two vertices or placing a vertex on an edge, or more complicated as creating a "thick vertex".

5 Simplifying The Aspect Graph

5.1 Finite-Resolution Aspects

The plane-sweep algorithm returns a representative viewing direction for each region in the finite-resolution aspect graph. For each such viewing direction we render the object as seen from that direction. We attempt to merge close features to generate a simpler view of the object, which we call the finite-resolution aspect.

When merging features we can not merge them in an arbitrary order because we have to take into account the fact that "nearness" is not transitive. To illustrate this let us present a few examples. In Figure 14(a), two edges which are close to each other appear in the image as a thick edge. As the distance between the edges is less than ϵ the two edges are perceived as one line. In Figure 14(b) however, edge v is close to edge u and to edge w, but edges u and w are far from each other. The resulting image is a thick region which can not be simplified into a line as in the previous example, and we have to leave the black rectangle in the representation. If an edge is close to a vertex and transitivity of nearness is not violated the vertex is placed on the edge. however, in the example shown in Figure 14(c), vertex p is near edges e_1 and e_2 that are adjacent to vertex q. The result should be merging p and q, but p is not near q, violating the transitivity of nearness. 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.

After simplifying the aspects of all the regions in the aspect graph, we compare aspects of neighboring regions, and regions with identical finite-resolution 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. Neigh-

4.4 Constructing the Regions and Aspects

We compute the regions from their bounding critical curves and curve intersections. For each face on the viewing cube we use a plane-sweep algorithm [16]. One of the important reasons for using the viewing cube instead of the viewing sphere was to enable us to use the standard plane-sweep algorithm on the faces of the cube. The algorithm was adapted for dealing with curves by adding critical events at extremal points, i.e., places where the curve is perpendicular to the sweeping direction [6]. Points where curves start, end or intersect are processed in sweep order, generating the non-critical regions of the aspect graph (Figure 13). For each region the algorithm returns a point in it which is its representative viewing direction.

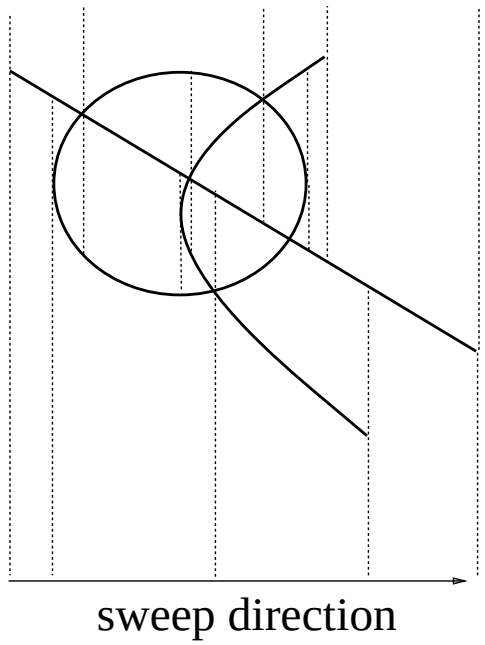

Figure 13: Illustration of the plane-sweep algorithm.

but features adjacent to the occluded feature are visible.

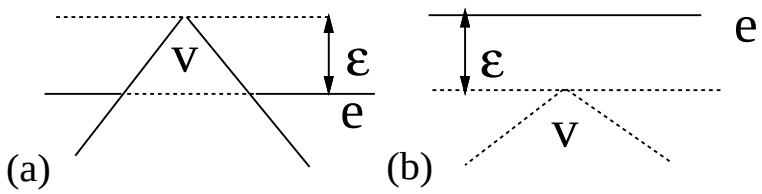

Figure 12: Occlusion of finite-resolution events: (a) although the point on the edge is occluded the event is considered visible because the other parts of the edge are visible; (c) the vertex and its adjacent edges are occluded therefore the event is not part of the critical curve.

A number of tests must be performed to check for occlusion. First, the features 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 features. 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 which share two of the three edges.

Second, the features participating in the event can also be occluded by faces adjacent to them. 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. So in all cases the occluded part of a curve starts and ends at intersection points with other curves, and we use the technique described above to recover these points.

compute a viewing direction for discrete values of the parameter, then normalize the viewing directions such that $|\mathbf{v}|_\infty = 1$.

4.2 Finding Intersections

We find intersection points between two curves using homotopy continuation. 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 cube, therefore we run the algorithm six times, each time setting one of the coordinates of $\mathbf{v}$ to ± 1, reducing the number of unknowns by one. The algorithm returns the intersection points. Points such that $|\mathbf{v}|_\infty > 1$ are discarded. We compute the parameter value for the point and discard points whose value is out of range. We insert the remaining points into the correct place on the curve, using their parameter value.

4.3 Occluded Events

Until now, a viewing direction has been considered to be part of a critical curve when the features (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 features must all be visible. This requirement has a straightforward meaning for infinite-resolution events but for finite-resolution events a more delicate definition is needed. Consider the example of an EV event in Figure 12(a). The position on the edge which is at distance ϵ away from the vertex is not visible but as we can see other parts of the edge, we consider this viewing direction as part of the critical curve. On the other hand in Figure 12(b), the vertex and its adjacent edges are occluded therefore this event is not part of the critical curve. Therefore we also consider a point on the critical curve visible, even when some of the features occlude each other

and (1) we recover $\mathbf{v}(t)$ using homotopy continuation [12]. An alternative method for finding $\mathbf{v}(t)$ is to find ϵ' which satisfies (3) using standard numerical one-dimensional optimization techniques such as a combination of the secant method and the Newton-Raphson method [8]. The method requires that for a given value of ϵ' we evaluate the function being minimized (3). We substitute the value of ϵ' into (1), which yield two linear equations in $\mathbf{v}(t)$, which are readily solved. We then evaluate (3) using the result. This method is more efficient then the previous one because instead of solving for ϵ' and $\mathbf{v}(t)$ at the same time, we use efficient one-dimensional optimization techniques to solve only for ϵ' and use its value to compute $\mathbf{v}(t)$.

4 Constructing the Aspect Graph

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. Eliminating 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 by sampling the parameters. We

through $p + (t + \epsilon'/|\mathbf{u}|)\mathbf{u}$ with direction $\mathbf{v}$ intersects the supporting line $D_2 = (a_2, b_2)$ of e_2 (see Figure 11). The line D_2' has Plücker coordinates $(\mathbf{v}, (t + \epsilon'/|\mathbf{u}|)\mathbf{u} \times \mathbf{v})$. Writing that D_1' (resp. D_2') intersects D_1 (resp. D_2), we obtain two equations:

$$\begin{cases} ta_1 \cdot (\mathbf{u} \times \mathbf{v}) + b_1 \cdot \mathbf{v} = 0, \\ (t + \epsilon'/|\mathbf{u}|)a_2 \cdot (\mathbf{u} \times \mathbf{v}) + b_2 \cdot \mathbf{v} = 0, \end{cases} \tag{1}$$

and eliminating t among these yields:

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

where ϵ' is the actual distance between the two intersection points on e. Squaring this equation yields a homogeneous equation of degree 6 in $\mathbf{v}$.

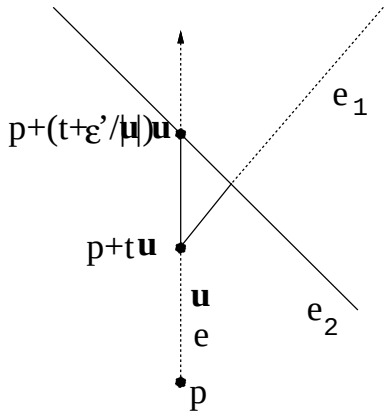

Figure 11: Illustration of the construction for the finite-resolution EEE event equation: The projection of points $e(t) = p + t\mathbf{u}$ and $p + (t + \epsilon'/|\mathbf{u}|)\mathbf{u}$ intersect the projections of e_1 and e_2 respectively.

We parameterize the curve using the parameter t used in (1). At viewing direction $\mathbf{v}(t)$ the point $e(t)$ on e intersects the projection of e_1, and the point $e(t) + (\epsilon'/|\mathbf{u}|)\mathbf{u}$ intersects the projection of e_2. Using

$$\epsilon' = \epsilon \frac{|\mathbf{v}(t)||\mathbf{u}|}{|\mathbf{u} \times \mathbf{v}(t)|}, \tag{3}$$

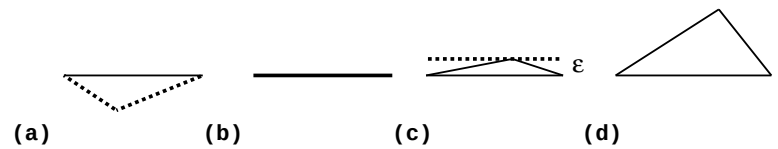

Figure 9: (a) The face is not visible. (b) Infinite-resolution face-appearing event. (c) The finite-resolution face-appearing event. (d) Face completely visible.

distance between the intersection points of the corresponding edges e_1 and e_2 with 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 (Figure 10). We now characterize the views $\mathbf{v}$ 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.

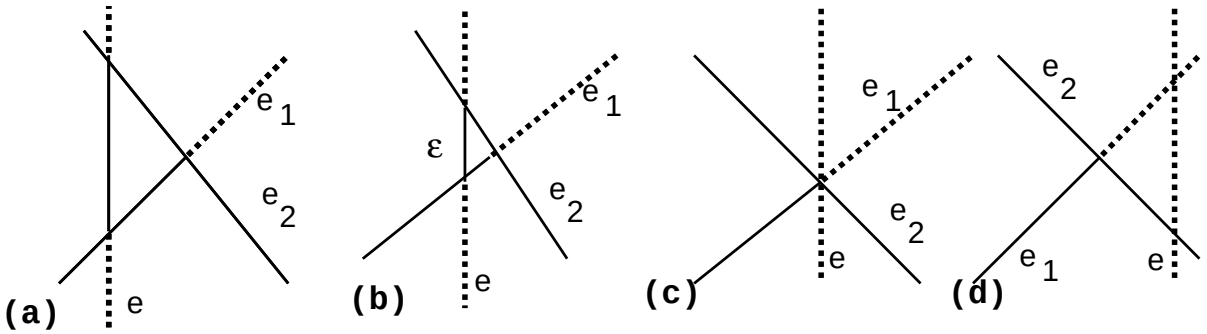

Figure 10: Finite-Resolution EEE event: (a) the three edges are visible; (b) the finite-resolution EEE event; (c) the infinite-resolution EEE event; (d) edge completely occluded.

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 $a_1 \cdot b_2 + a_2 \cdot b_1 = 0$. We write that a line D_1' passing through $e(t) = p + t\mathbf{u}$ with direction $\mathbf{v}$ 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 $(\mathbf{v}, t\mathbf{u} \times \mathbf{v})$. Similarly, we write that a line D_2' passing

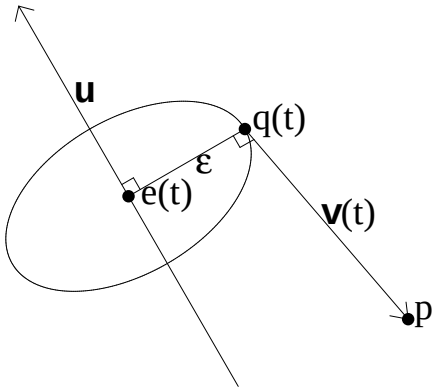

Figure 7: Parametric construction of the EV curve. Illustration of the geometric construction of the viewing direction $\mathbf{v}(t)$ corresponding to the point $e(t)$ on the edge.

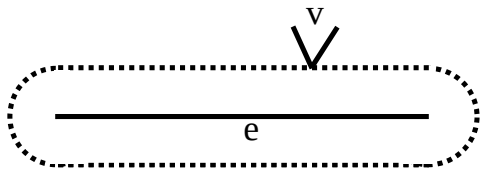

Figure 8: The positions where the projection of the vertex can be with respect to the edge.

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 (Figure 9).

3.3 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

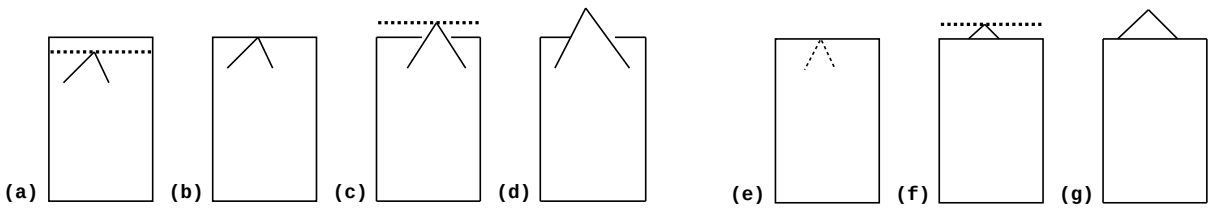

Figure 6: finite-resolution EV event. When the vertex is in front of the edge: (a) first finite-resolution EV event; (b) infinite-resolution EV event; (c) second finite-resolution EV event; (d) vertex completely above the edge. When the edge is in front of the vertex: (e) infinite-resolution EV event; (f) finite-resolution EV event; (g) vertex completely above the edge.

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 $\mathbf{v}(t)$ by finding a point $q(t)$ such that $\mathbf{v}(t) = p - q(t)$. $q(t)$ satisfies the following conditions which are illustrated in Figure 7:

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

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 $\mathbf{u}$ 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 semi-circle (Figure 8). This semi-circle is part of the curve of the VV event we discussed in the previous section.

It should also be noted that, for infinite-resolution aspect graphs, EV events for which

and $\mathbf{u}$ is $\theta = \arcsin(\epsilon/|\mathbf{u}|)$. They form opposite small circles on the viewing sphere, which can be parameterized as follows: for each angle $0 \leq \alpha \leq 2\pi$ we compute a point that when subtracted from p produces the desired viewing direction. As shown in Fig 5, a parameterization of one of these circles is :

$$\mathbf{v}(\alpha) = \mathbf{u} + d(\cos\alpha\mathbf{u}_2 + \sin\alpha\mathbf{u}_3) \quad \alpha \in [0, 2\pi],$$

where $\mathbf{u}, \mathbf{u}_2, \mathbf{u}_3$ is an orthonormal basis of $\mathbb{R}^3$ and $d = \mathbf{u}\tan\theta$. The opposite circle is $-\mathbf{v}(\alpha)$.

3.2 Edge-Vertex (EV) Event

With infinite-resolution cameras, 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 (Figure 6). 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 (Figure 6(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 (Figure 6(e-g)).

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

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

which is a quadratic equation in $\mathbf{v}$.

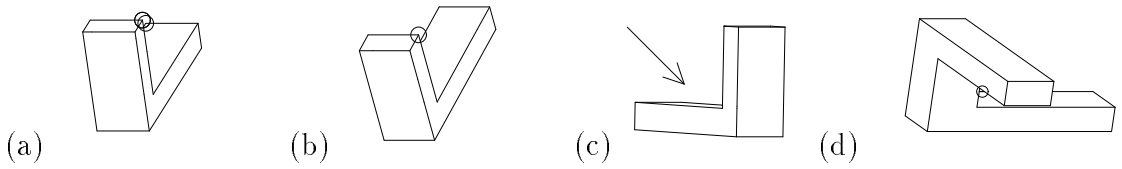

Figure 4: Four visual events: (a) a VV event; (b) an EV event; (c) a face begins to appear; (d) a EEE event.

between the projections of the vertices, until it exceeds the camera resolution ϵ and the vertices are seen as distinct. The visual event occurs when the image distance is exactly ϵ. The same phenomenon happens for a viewing direction parallel to the direction of an edge. At that viewing direction, both ends of the edge coincide and the edge is foreshortened to zero. When the viewing direction changes and the length of the projected edge is more than ϵ, the edge becomes visible. In both cases, when the distance between the projections of the vertices is less than ϵ, they appear to coincide in the image and the adjacent edges seem to emanate from the same point.

Let $\mathbf{v}$ denote the viewing direction. The vertices p and q are represented by the pair $(p, \mathbf{u})$, where $q = p + \mathbf{u}$. The vertices are observable as distinct vertices if and only if $|\mathbf{u} \times \mathbf{v}| \geq \epsilon |\mathbf{v}|$ (Figure 5).

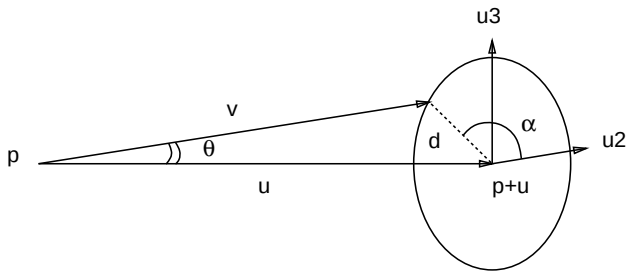

Figure 5: Illustration of the VV event.

The corresponding visual events correspond to views such that the angle between $\mathbf{v}$

foreshortening of an edge 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. We will partition the viewing sphere into non-critical regions bounded by critical curves using a plane-sweep algorithm [16]. For each region a representative view of the object is handed to a simplification algorithm, which merges close features generating the finite-resolution view of the object. Neighboring regions with identical finite-resolution views are merged producing the finite-resolution aspect graph.

3 Finite-Resolution 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 4 shows the four new visual events. We derive implicit equations and a parametric representation for these curves. The corresponding viewing directions are parameterized such that each viewing direction has the same parameter value as a corresponding point on an edge. For example for the EV curve each point on the edge has the same parameter value as the viewing direction on the curve where the projection of the vertex is closest to the projection of that point. The implicit equations are used to find intersection points between curves and the parametric representations are used to trace curves, find end-points on these curves corresponding to end-points of the edge, and order the points on each curve by their parameter value.

3.1 Vertex-Vertex (VV) Event

When two vertices are viewed from a direction parallel to the line joining them, their projections coincide in the image. Changing the viewing direction increases the distance

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, 15, 23]: 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 (Figure 2). In the perspective projection model, every point in an infinite 3D space represents a possible viewpoint. In this model, points from which the critical events are viewed form 2D surfaces. Under the orthographic projection model we assume that we view the object at infinite distance. Thus the viewing rays can be considered as points on a viewing sphere at infinity. The intersection of that sphere with the surfaces of the critical events yields a set of critical curves. Consider the example of Figure 3: for a certain viewing direction a vertex is projected on one side of an edge (Figure 3(a)). As the viewing direction continually changes the projection of the vertex will coincide with the projection of the edge (Figure 3(b)) and later be projected on the other side of the edge (Figure 3(c)). For each point on the edge there is a viewing direction for which its projection and the projection of the vertex coincide. Combining these viewing directions for all the points on the edge yields a critical curve.

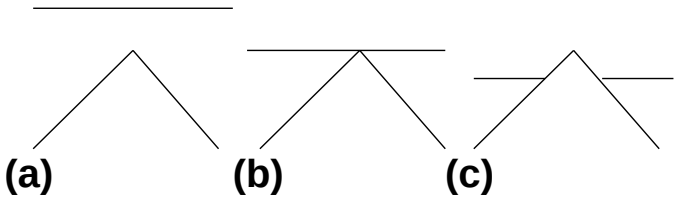

Figure 3: An EV event: (a) before the event; (b) at the event (c) after the event.

By taking resolution into account, we will introduce new types of visual events related to EV and EEE events. They will correspond to two vertices being so close they seem to coincide, a vertex seeming to touch an edge (similar to an EV event), and to

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 fifteen 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, 15, 19] and perspective [22, 23, 24, 25] projection. Some of these algorithms have actually been implemented (e.g., [15, 19, 22, 23, 24, 25]). 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, 14, 17, 21] have also been proposed and several of them have also 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 their projections in the image is greater than ϵ.

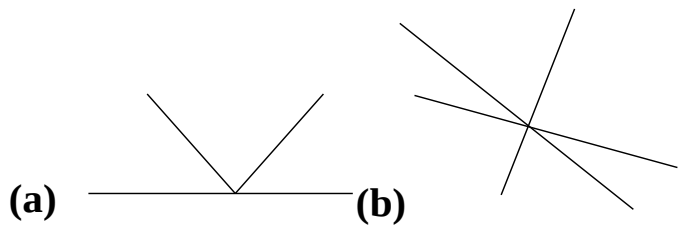

Figure 2: Infinite-resolution visual events: (a) an EV event; (b) a EEE event.

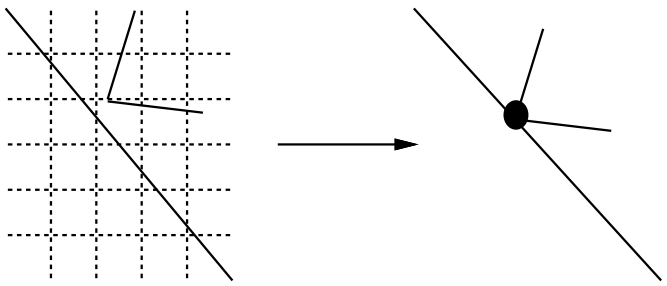

Figure 1: The effect of finite-resolution on an image. An edge and a vertex which do not touch each other in the infinite-resolution image seem to touch each other when finite-resolution is assumed.

The rest of the paper is organized as follows. Section 2 discusses previous methods for aspect graph computation and introduces our approach. In Section 3, we present a catalogue of visual events adapted to our imaging model. It is an extension of the familiar EV-EEE events used by others. We then present in Section 4 an algorithm for computing the finite-resolution aspect graph of a polyhedron. An algorithm for simplifying the aspects by merging regions with identical finite-resolution aspect graphs is presented in Section 5. We discuss the size of the finite-resolution aspect graph and the complexity of the algorithm in Section 6. The implementation of the algorithm and results are presented in Section 7. Finally, a number of issues raised by our algorithm and its implementation are discussed in Section 8.

2 Literature Review

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

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, 14, 18]. The time complexity of the algorithms for constructing the aspect graph is at least as high [6, 7, 14, 15]. In practice, this means that one can only use exact aspect graphs of relatively simple objects in tasks such as object recognition. However by taking resolution into account we can hopefully 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 classical 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. The effect of resolution is illustrated in Figure 1 where the distance between an edge and a vertex is below the resolution of the camera, and so these features seem to touch in the finite-resolution image.

Finite-Resolution Aspect Graphs of Polyhedral Objects

Ilan Shimshoni* and Jean Ponce
Department of Computer Science and Beckman Institute
University of Illinois, Urbana, IL 61801

January 16, 1996

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

We address the problem of taking image resolution into account during the construction of the aspect graph [11] of a polyhedral object. 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

*Ilan Shimshoni is currently at the Computer Science Department, Technion, Haifa 32000, Israel, e-mail: ilan@cs.technion.ac.il