# 3D Models from Occluding Contours using Geometric Primitives *

Morten Bro-Nielsen
Institute of Mathematical Statistics and Operations Research (IMSOR)
The Technical University of Denmark (DTH)

September 30, 1994

## Abstract

This paper proposes a new method for generating complete 3D models of complex objects using the occluding contour and perspective projection. Using a known motion around the object, such as an epipolar movement, the occluding contour is registered from a number of viewpoints. When the occluding boundaries from different viewpoints are projected into space they determine an enclosing volume for the object. The contribution of this paper is a new method for generating a polygonal 3D model surface directly from the projected occluding boundaries. A projected occluding contour is represented by triangles. Quadrangles, that are subparts of the triangles, are extracted to approximate the shape of the object. The important difference of this method compared to previously published similar approaches is that it produces a polygonal surface directly with no intermediate steps such as a voxel representation. By assuming orthogonal instead of perspective projection and epipolar movement of the camera, a simpler method is also derived that uses 2D lines in planes instead of 3D triangles in space. Finally texture mapping is demonstrated.

## 1 Introduction

Numerous methods for acquiring 3D models of real objects have been proposed and implemented in the last decade. These methods can roughly be divided into two distinct groups based on whether they acquire 3D shape information using an active or a passive sensor. Active sensors use some form of radiation to generate a reflection from the object surface which is used to determine shape parameters. Surveys of different active methods are given in [3] and [16]. Passive sensors use images of the objects to determine the shape using a variety of shape from ... methods.

Generally, active sensors give the best results and most commercially available systems also employ these. But active sensors are expensive and are not practical for widespread use e.g. as multi-media components. Much work is therefore focussing on developing passive methods for acquiring 3D models. One of the many shape from ... methods that are investigated for this purpose is shape from the occluding contour.

The occluding contour (or occluding contour, silhouette, profile, apparent contour) is the projection of the extremal boundary (or contour generator, rim) of the object onto the image plane (see figure 1). At the extremal boundary the visual direction is a tangent to the object surface.

It has been shown how the deformation of the occluding contour under motion can be used to determine the local 3D shape at the extremal boundary of the object using differential geometry ([11, 12, 13, 17]). But this approach is sensitive to abrupt changes in surface direction and irregular shapes, and has yet to be demonstrated for making complete models of complex objects.

Another approach to 3D shape acquisition uses the fact that the occluding contour determines a bounded volume in space for the object. An occluding contour from one direction creates a cone of tangents to the extremal boundary that go through the viewpoint. This cone determines an open bounding volume. By using two or more views the combined bounding volume becomes closed. An approximation to the object shape is determined as the minimal bounding volume of the combined cones. Increasing the number of views increases the precision of the resulting 3D model.

This kind of approach can handle arbitrary shapes, provided that the object surface does not contain concavities or "holes". As the shape is determined by tangents to the surface, these shape primitives are not visible in the input data.

The employed algorithm can be described by the nature of the primary data structure (volume or surface based - see [1] and [22] for a discussion) and the type of projection (orthogonal or perspective). Volume rep-

*This work was carried out during a 6 month period, when the author was a trainee at the Kansai Research Laboratory of Toshiba Corporation in Japan.

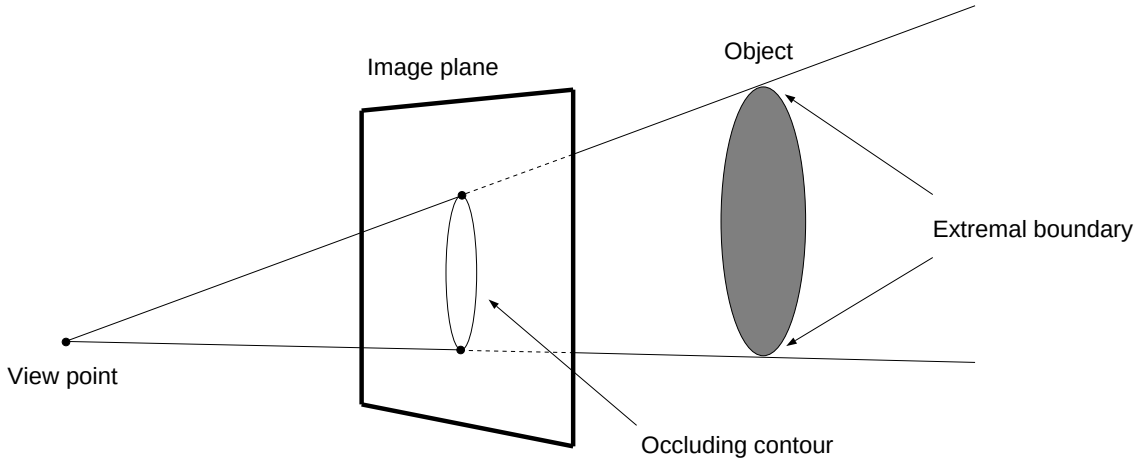


Figure 1: The extremal boundary is projected onto the image plane as an occluding contour.

resentations using orthogonal or perspective projections seems to be popular ([2, 6, 7, 8, 9, 10, 15, 18, 20]). Unfortunately volume representations tend to be memory consuming and usually have to be converted to surface representations for graphics display. Developing surface representations directly can therefore be attractive both from a memory and application point of view. Only [19] have described work that in part developed a surface representation directly. But this was based on orthogonal projection which simplifies the problem considerably. Because methods that use orthogonal projection generally use image material provided by a perspective camera, the orthogonal projection is an assumption which induces inaccuracies in the shape depending on the credibility of the assumption. It was shown in [23] that the ratio between the distance from the camera and the height of the object must be higher than 10 for the assumption to be reasonable. Of course this creates dilemmas when modelling even small objects.

The contribution of this paper is a new method for generating a surface representation directly using perspective projection.

The method uses 3D triangles as primitives in the bounding cone generated by the occluding contour in an image. The arcs in a polygonal approximation to the occluding contour is used to generate the individual triangles. The first step of the algorithm registers how triangles from different views collide. The second step uses this information to determine quadrangles that combine to produce the surface of the minimal bounding volume. Finally, texture is mapped onto the resulting model.

The paper also derives a simpler version based on the dual assumptions of orthogonal projection and epipolar movement, which is somewhat similar to part of [19]. The simple version uses 2D lines in planes instead of 3D triangles in space and thereby reduces the complexity considerably.

# 2 3D Model Generation

This section describes how the model is produced from occluding boundaries determined in images captured from different viewpoints. It starts by describing how triangles are produced based on the occluding contour, then shows how triangle collisions are detected and finally shows how the final shape is extracted from the triangles as quadrangles.

## 2.1 Determining the Occluding Contour

The occluding contour have been determined by recording silhouette images of the object using a white illuminated backscreen. A blue background was also used, but the silhouette images provided the more robust results. The occluding contour is extracted from silhouette images by thresholding these and then registering the boundary of the resulting object region. The boundary is represented using a polygonal approximation computed using an algorithm described in [21]. The polygon is directed counterclockwise if the polygon surrounds an object region and clockwise if the surrounded region is space.

## 2.2 From Occluding Contours to Triangles

An occluding contour is represented as a directed closed polygon in an image. By connecting the end points of each polygon arc to the viewpoint of the camera each polygon arc determines a triangle in space (see figure 2). The polygon arcs of a closed contour create a closed

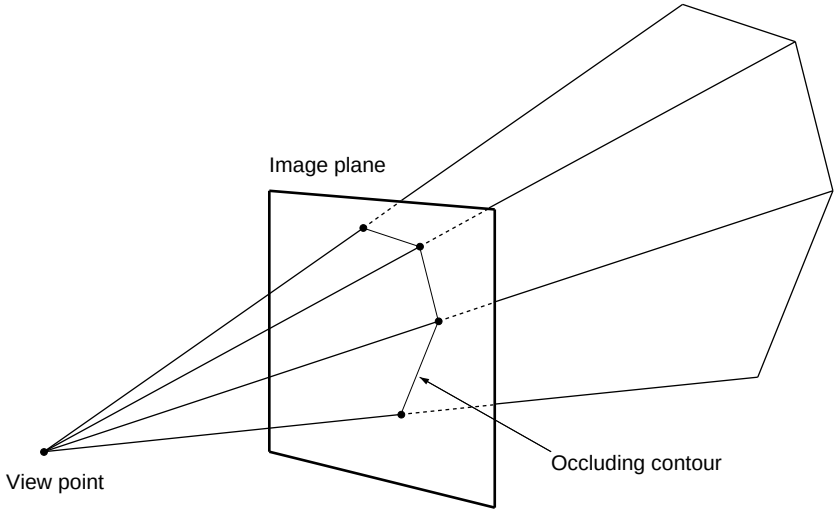


Figure 2: Contour polygon arcs define triangles in space.

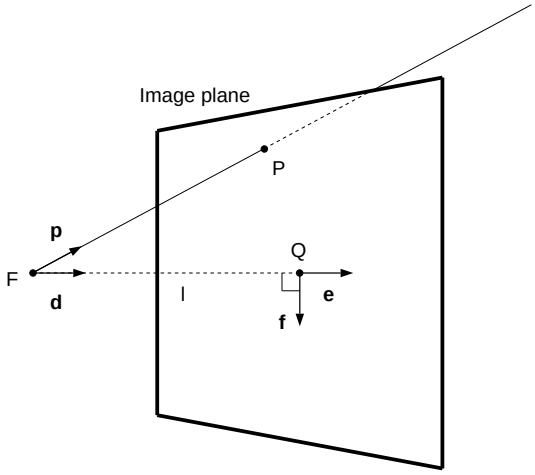


Figure 3: Camera model.

cone of triangles and by projecting these triangles into space a bounding volume for the object is produced. Because of the polygon direction the normals of the triangles points towards space, assuming that the coordinate system obeys the right hand rule.

The corner points of a triangle defined by a polygon arc are determined using the geometric relationship between the viewpoint and the image points. The camera model is shown in figure 3. The direction of the camera is described by $\boldsymbol{d}$ and the view or focus point is $F$. The image plane defined by the basis $(\boldsymbol{e}, \boldsymbol{f})$ is placed in front of the focus point perpendicular to the camera direction and the distance $l$ from $F$. The center of the image plane is $Q$. Notice that $(\boldsymbol{e}, \boldsymbol{f}, \boldsymbol{d})$ forms a right handed 3D coordinate system with $Q$ as origin.

Together with the viewpoint an image point $P = (x, y)$ defines a line in space with parametric origin $F$ and directional vector:

$$\boldsymbol{p} = \frac{\boldsymbol{FP}}{|FP|} = \frac{x \times \boldsymbol{e} + y \times \boldsymbol{f} + l \times \boldsymbol{d}}{\sqrt{x^2 + y^2 + l^2}} \tag{1}$$

The line can, therefore, be described in the parametric form:

$$\boldsymbol{l}(t) = \boldsymbol{OF} + t \times \boldsymbol{p} \tag{2}$$

Consequently, a triangle projected from a directed polygon arc $Arc = [P_1, P_2]$ has the form

$$Tri(Arc) = [F, \boldsymbol{l}_1(t_{max})^*, \boldsymbol{l}_2(t_{max})^*] \tag{3}$$

where $t_{max}$ is the length of the long triangle arcs and $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$ are the lines defined by $P_1$ and $P_2$ respectively.

Observe that an observation of an object typically consists of a number of views, each containing a number of closed contours which produce closed cones and each cone consisting of a directed list of triangles[1]:

$$\begin{aligned} Observation &= \{View_1, View_2, ...\} \\ View_i &= \{Cone_{i1}, Cone_{i2}, ...\} \\ Cone_{ij} &= < Tri_{ij1}, Tri_{ij2}, ... > \end{aligned} \tag{4}$$

## 2.3 Registering Triangle Collisions

Triangles created from the occluding contours of one view only touch each other along the long edges where they are connected into cones. They do not cross as a contour cannot cross another contour from the same view. When more than one view of the same object is

[1] $< ... >$ is used to denote an ordered set. $\{ ... \}$ denotes a normal unordered set.

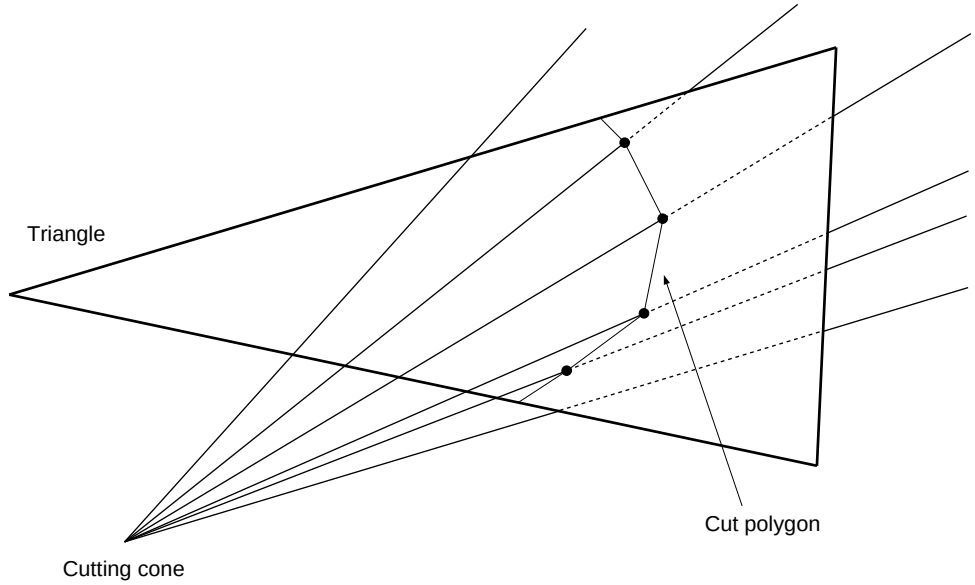


Figure 4: A cut polygon is generated by a colliding triangle cone.

used, triangle cones from one view can and will collide with cones from other views.

If a triangle is hit by a triangle cone $Cone_{ij}$ from another view $View_i$, this cone will create a *cut* polygon

$$Cut_{ij} =< Arc_1, Arc_2, ... > \tag{5}$$

in the triangle, where each arc in the polygon is generated by a triangle in the cutting cone. The situation is illustrated in figure 4, where a cone generates a cut polygon in a triangle.

## 2.4 Extracting Quadrangles

The set of cones from a view $View_i$ defines an open bounding volume for the object and this bounding volume is projected onto the triangle as a set of cut polygons

$$Cuts_i = \{Cut_{ij}\} \tag{6}$$

This set, therefore, defines a partitioning of the triangle into two non-overlapping sets:

$$\begin{aligned} Tri &= Tri^{out}(Cuts_i) \cup Tri^{in}(Cuts_i) \\ \wedge \emptyset &= Tri^{out}(Cuts_i) \cap Tri^{in}(Cuts_i) \end{aligned} \tag{7}$$

where $Tri^{out}(Cuts_i)$ is outside the bounding volume of $View_i$ and $Tri^{in}(Cuts_i)$ is inside. A triangle will usually be cut by sets of cones from many views defining many partitions on the triangle.

To determine the minimal bounding volume for the object it is necessary to determine the parts of all triangles that are closest to the object. If a particular part of a triangle from one view is closest, it must be inside a bounding volume of each of all the other views as there otherwise would exist a triangle from another view with parts closer to the object. As the bounding volume of each $View_i$ is defined on the triangle by the above mentioned sets, the objective is therefore to determine the common set:

$$Tri^{min} = \begin{cases} \emptyset & \|C\| < n-1 \\ \bigcap_{i \epsilon C} Tri^{in}(Cuts_i) & \|C\| = n-1 \end{cases} \tag{8}$$

where $C$ is the set of views for which at least one cone cut the triangle. $n$ is the total number of views.

To determine $Tri^{min}$ for a triangle, the triangle is first divided into slices defined by the nodes in the cut polygons (cut nodes) and collision points between pairs of polygon arcs (collision nodes) (see figure 5). Together with the viewpoint of the triangle each node defines a line in the triangle. These lines partition the triangle into a number of slices.

The slices have the nice property that the polygon arcs inside each slice have a fixed ordering. In figure 5 one such slice is extracted for analysis in the bottom of the figure.

In each slice, pairs of adjacent cut polygon arcs define quadrangular patches (quadrangles). The cut polygon arcs define the open bounding volumes on the triangle/the slice, and these open bounding volumes alone determine the object shape/the minimal bounding volume. A quadrangle is therefore an atomic surface primitive.

To extract quadrangles that are part of the minimal bounding volume, the quadrangles are traversed

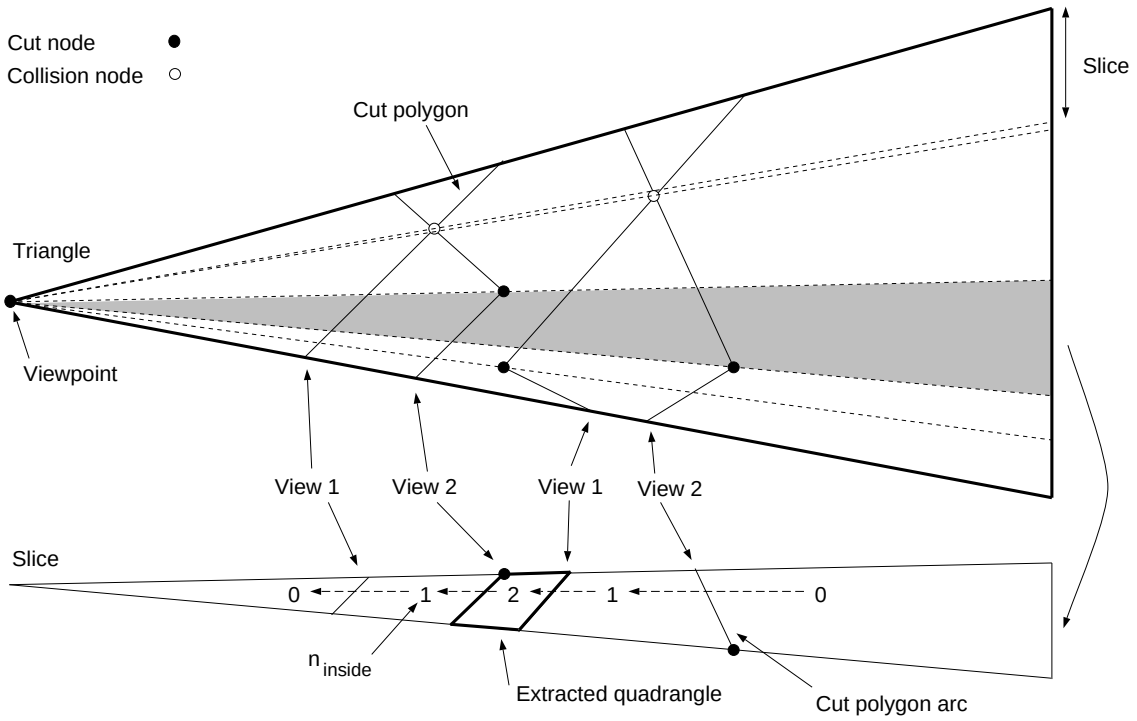


Figure 5: Top: A triangle is partitioned by cut polygons. Bottom: Resulting quadrangle is extracted from a triangle slice.

from one end of the slice towards the other (see figure 5/bottom). The views that the current quadrangle is inside is determined using a binary status variable $inside_i$ where $i$ is the view number. If the current quadrangle is inside a cone from $View_i$ then $inside_i = 1$. Otherwise $inside_i = 0$.

A polygon arc that separates two adjacent quadrangles determines the boundary of a cone from exactly one $View_i$. Therefore, crossing this arc during the traversal alone changes the status variable corresponding to this view:

$$inside_i = \neg inside_i \tag{9}$$

The number of active views $n_{inside}$ for a quadrangle can be determined simply by adding the current status variables:

$$n_{inside} = \sum_i inside_i \tag{10}$$

In the example in figure 5 the traversal goes from right to left in the slice. At first $n_{inside}$ is 0. As the traversal crosses the first polygon arc generated by a cone from $View_2$, and thereby enters into this view, it changes to 1 and so on.

For a quadrangle to be part of the minimal bounding volume it must be inside a cone from all views except its own according to (8). Or in other words that:

$$n_{inside} = n - 1 \tag{11}$$

where $n$ is the total number of views. Quadrangles that satisfy this requirement are therefore extracted as part of the minimal bounding volume.

In figure 5 the number of views is $n = 3$. Therefore, the quadrangle with $n_{inside} = n - 1 = 2$ is extracted as a part of the minimal bounding volume.

The extracted quadrangles of a triangle form the common set $Tri^{min}$ of equation (8). The minimal bounding volume is the combination of all $Tri^{min}$.

# 3 Simplification using Orthogonal Projection

In this section the problem is restricted by two assumptions: Orthogonal projection and epipolar movement. These assumptions simplify the problem considerably and reduces the algorithmic complexity.

The epipolar movement of the camera means that every view is registered in the same plane with the direction vector $\boldsymbol{d}$ of the camera pointed towards the same point and parallel to the plane. In this case the same

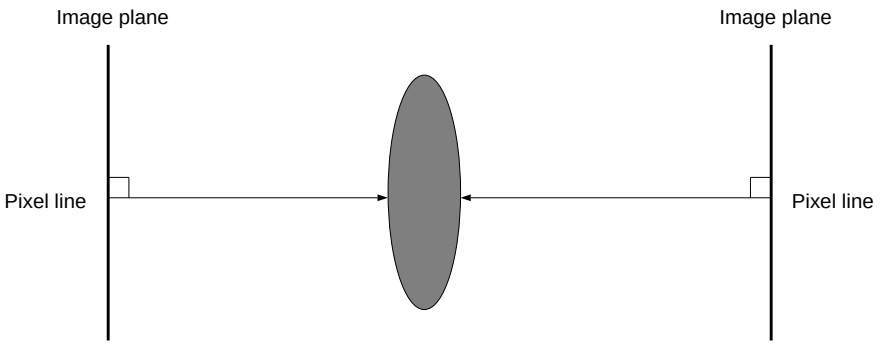


Figure 6: When orthogonal projection and epipolar movement is used, the same pixel lines in images from different views lie in the same plane.

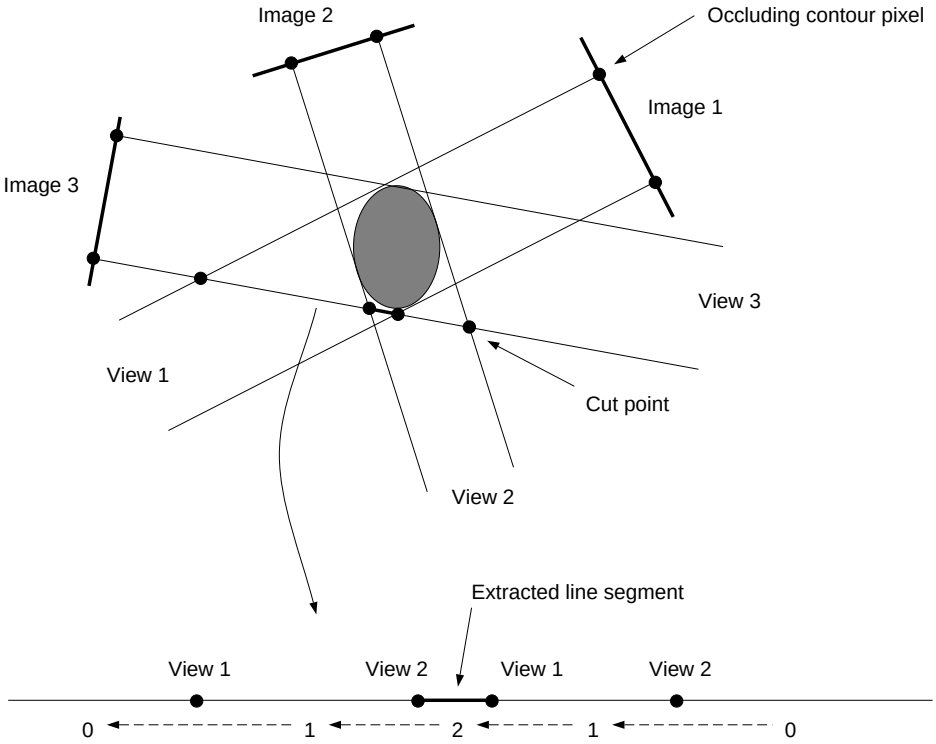


Figure 7: Top: Lines generated by occluding contour pixels from different views cut each other in cut points. Bottom: Resulting line segment is extracted.

image pixel lines of all the images taken at the viewpoints will be in exactly the same plane (see figure 6). This is an important observation because it means that the shape of the part of the object model that intersects such a plane is influenced by information in this plane alone. Consequently, each plane can be handled by itself. This reduces the problem from 3D as in the last section to 2D. Instead of handling triangles in 3D space it is only necessary to handle lines in 2D planes.

Consider one of the planes defined by a pixel line in the view images. In such a plane each view image is represented by a pixel line and the occluding contour as single pixels (see figure 7). Each occluding contour pixel defines a line that splits the plane into two sub-planes of which one contains the object and the other is space. This line corresponds to a triangle in the perspective case. Lines from the same view cannot cross each other. But each line is cut by lines from other views. A list of cut points is therefore defined for each line. The cut points correspond to the cut polygons in the perspective case.

Every pair of adjacent cut points defines a line seg-

ment for which the considerations concerning the minimal bounding volume in the last section are valid. To extract the line segments that are part of the minimal bounding volume the line is traversed in the same way as the quadrangles were traversed in the last section. For each line segment the number $n_{inside}$ of views that it is inside is determined using the status variable $inside_i$ and if the number is equal to the number of views minus one, the line segment is extracted as part of the minimal bounding volume. By performing this algorithm for all lines in all planes the object shape is determined.

This simplified version of the general perspective algorithm, using the assumptions of orthogonal projection and epipolar movement, is similar to part of the work presented in [19], although the apparent principles are different.

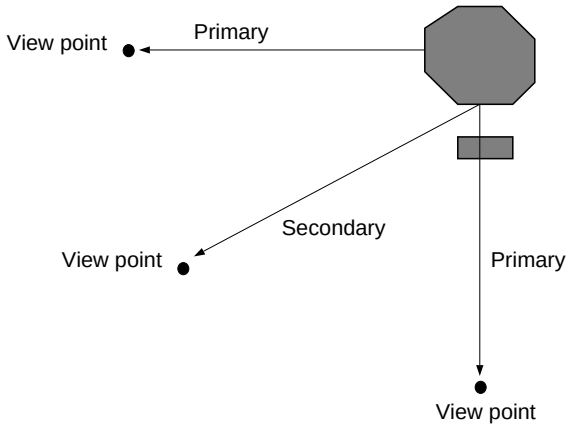


Figure 8: Choice of view for texture mapping.

# 4 Texture Mapping

To enhance the result of the perspective algorithm, texture mapping is used to give the generated 3D model a natural looking surface. Colour images taken at the viewpoints are used as texture for the quadrangles. When many viewpoints are used, the best viewpoint for each quadrangle must be selected. Basicly two properties must be satisfied to ensure the best texture:

- The viewpoint direction from a quadrangle should be as close to the quadrangle normal as possible.
- The viewpoint should be visible from the quadrangle.

The first requirement ensures that textures from extreme sideviews are not used. The second requirement ensures that the texture is actually originating from the part of the surface that the quadrangle models. Figure 8 shows the meaning of the two requirements. The first is satisfied by the primary viewpoints. The second is not satisfied by one of the primary viewpoint and a secondary choice is used.

Computationally, the first requirement is easy to satisfy, by ordering the viewpoints for each quadrangle and selecting the best. The second is considerably more difficult as it can only be satisfied by checking all the resulting quadrangles against each other for each viewpoint. Several tricks can be used to reduce the complexity, but it remains high.

# 5 Results and Discussion

The perspective and orthogonal algorithms have been used to model a number of different objects. Figure 10 shows the result of applying the perspective algorithm to a cup. Figure 11 shows the result for a little plastic doll. Also the shape of a human being has been modelled (figure 12). Unfortunately, the model moved his head during the recording session which resulted in a compression of the head.

All these examples were recorded using an epipolar movement of the camera. Generally, the perspective algorithm allows for any combination of viewpoints except cases where two triangles from different views are in the same plane. This can occur if the line between two viewpoints is perpendicular to the parallel viewing directions of the two viewpoints. When two triangles are in the same plane, it is impossible to determine a cut polygon.

Complex cases could involve views that do not "see" each other. In this case the possible area that an image from a viewpoint can describe must be registered. This can be achieved very easily by including triangles determined by the edge of the camera/image. These special triangles indicate when a view is active and are

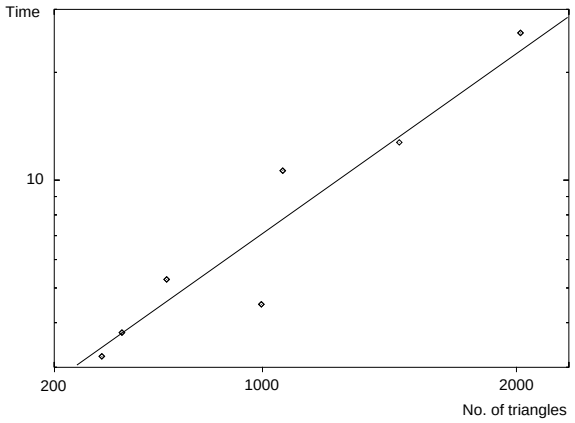


Figure 9: The complexity of the perspective algorithm is exponentially dependent on the number of triangles.


Figure 10: Cup modelled using perspective algorithm. 3D model creation took 23 minutes on a SiliconGraphics IRIS 4D/420VGX using 12 views with an average of 170 occluding contour polygon arcs in each view image. Upper row shows original. Bottom row shows model with and without texture.

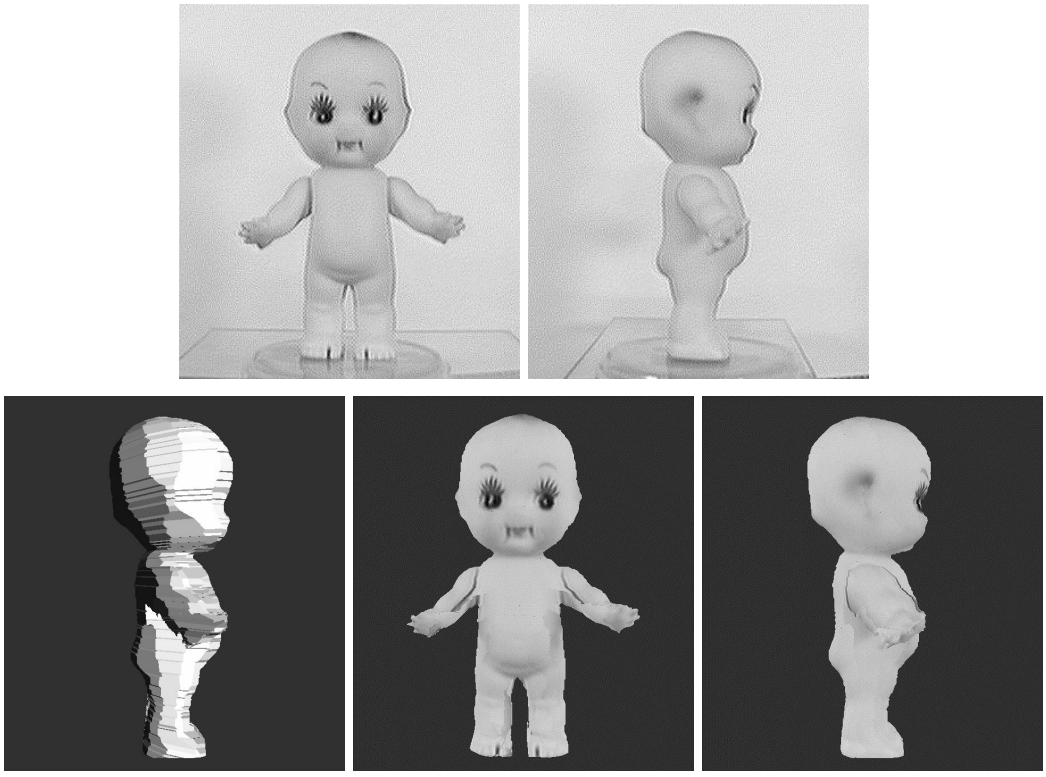

Figure 11: Doll modelled using perspective algorithm. 3D model creation took 46 minutes on a SiliconGraphics IRIS 4D/420VGX using 12 views with an average of 236 occluding contour polygon arcs in each view image. Upper row shows original. Bottom row shows model with and without texture.

Figure 12: Human being modelled using perspective algorithm. 3D model creation took 19 minutes on a Silicon-Graphics IRIS 4D/420VGX using 8 views with an average of 261 occluding contour polygon arcs in each view image. Left 2 images show original. Right 2 images show model with texture.

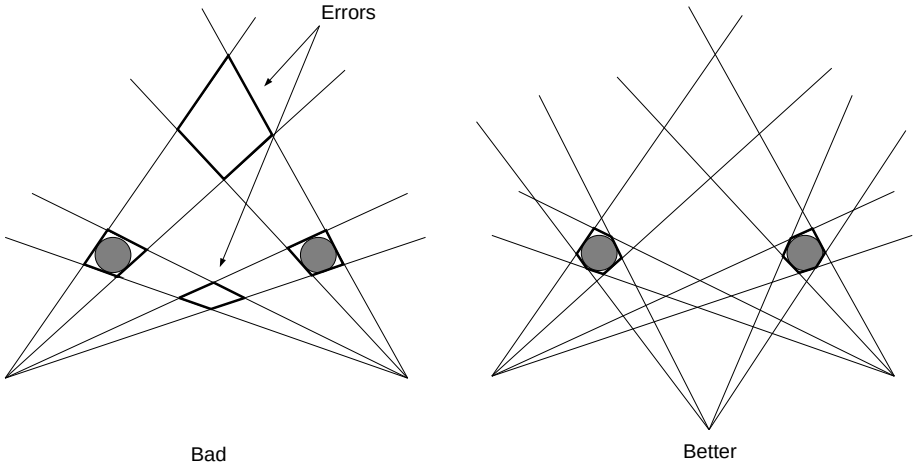


Figure 13: The choice of viewpoint positions is not trivial. The left example shows a typical problem occurring when a poor set of view points is used.

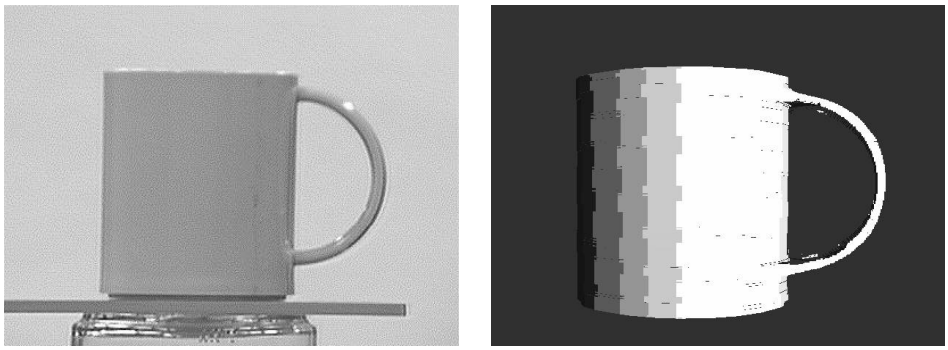

Figure 14: Cup modelled using orthogonal algorithm. 3D model creation took 1 minute on a SiliconGraphics IRIS 4D/420VGX using 12 views with 336 planes or pixel lines in each view image. Left image shows original. Right image shows model.

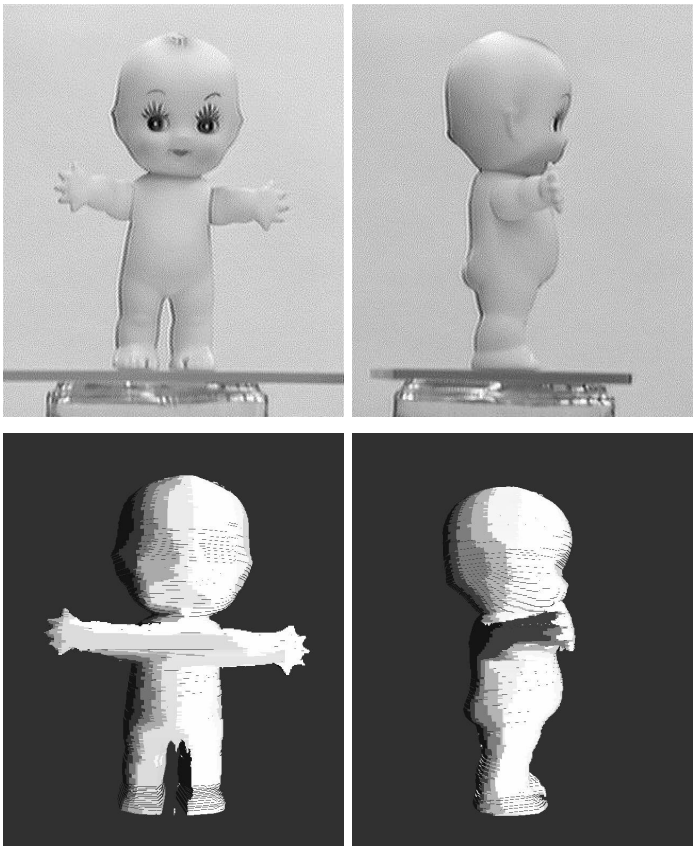

Figure 15: Doll modelled using orthogonal algorithm. 3D model creation took 1.5 minute on a SiliconGraphics IRIS 4D/420VGX using 12 views with 336 planes or pixel lines in each view image. Upper row shows original. Bottom row shows model.

processed just as the regular triangles.

Attempts at modelling an office environment using this extension to the perspective algorithm proved the general idea. But problems in getting enough views to make proper models ruined the result. The problem is illustrated in figure 13 where a bad combination of viewpoints generates two false objects. Adding a third viewpoint solves the problem in this case. But selecting the viewpoints is generally not a trivial task.

The results of using the orthogonal algorithm on a cup and a doll is shown in figure 14 and 15 respectively. The orthogonal algorithm is considerably faster than the perspective algorithm. But the viewpoints have to be relatively far from the object (10 times height - [23]) to give proper results. In the case shown in figure 14 the distance was one meter and modelling a human being as in figure 12 would require a distance larger than 17 meters. This limits the size of the possible objects and makes the setup rather big. Also remember that the orthogonal algorithm uses the assumption of epipolar movement of the camera contrasting the more free positioning of the viewpoints for the perspective algorithm.

Experiments show that the complexity of the perspective algorithm only is dependent on the number of initial triangles. Increasing the number of viewpoints generally just increases the number of triangles linearly and does not change the dependence of the complexity on the number of triangles. Figure 9 indicates that the complexity, unfortunately, is exponentially dependent on the number of initial triangles.

The number of initial triangles is determined by the number of views and the number of arcs in the polygons describing the occluding contour in each viewpoint image. Both factors influence the precision of the resulting model and have to be weighed against each other. For an epipolar movement the number of views tends to determine the precision of model intersection curves in planes parallel to the camera plane. The precision of the occluding contour polygon determines the precision of model intersection curves in planes perpendicular to the camera plane.

The complexity of the orthogonal algorithm is

$$m \times n(n-1)/2 \tag{12}$$

where $n$ is the number of lines generated by occluding contour pixels in each plane. $m$ is the number of planes.

# 6 Conclusion

In this paper a new method for modelling complex 3D objects based on the occluding contour has been presented. This method uses geometric primitives and, therefore, does not depend on volume representations that usually are used in similar methods. Volume representations can limit the precision of the resulting model. The new method is completely scalable and retains the precision that is possible from the input data.

A simpler method derived from the general method has also been developed. This method uses the dual assumptions of orthogonal projection and epipolar movement to reduce the complexity of the problem from handling 3D triangles in space to handling 2D lines in planes. It is therefore faster.

Examples have shown good results for the new method. Models of small objects such as a doll and a cup, and a large object such as a human being have been shown here.

# References


[1] Aggarwal, J.K., Davis, L.S., Martin, W.N. and Roach, J.W.: *Survey: Representation methods for three-dimensional objects*, in Kanal, L.N. and Rosenfeld, A.: *Progress in Pattern Recognition*, New-York: North-Holland, Vol. 1, pp. 377-391, 1981

[2] Aggarwal, J.K. and Chien, C.H.: *3-D structures from 2-D images*, in *Advances in Machine Vision*, Berlin: Springer-Verlag, pp. 64-121, 1989

[3] Besl, P.J.: *Active, Optical Range Imaging Sensors*, Machine Vision and Applications, No. 1, pp. 127-152, 1988

[4] Beusmans, J.N.H., Hoffman, D.D. and Bennett, B.M.: *Description of solid shape and its inference from occluding contours*, Journal of the Optical Society of America, Vol. 4, No. 7, pp. 1155-1167, 1987

[5] Brady, M., Ponce, J., Yuille A. and Asada, H.: *Describing surfaces*, Computer Graphics and Image Processing, Vol. 32, pp. 1-28, 1985

[6] Chien, C.H. and Aggarwal, J.K.: *A volume/surface octree representation*, Proceedings 7th International Conference on Pattern Recognition, Montreal, Vol. 2, pp. 817-820, 1984

[7] Chien, C.-H. and Aggarwal, J.K.: *Volume/surface octrees for the representation of 3-D objects*, Computer Vision, Graphics, and Image Processing, Vol. 36. pp. 100-113, 1986

[8] Chien, C.-H. and Aggarwal, J.K.: *Computation of volume/surface octrees from contours and silhouettes of multiple views*, Proceedings IEEE Conference on Computer Vision and Pattern Recognition,, Miami Beach, Fl. June 22-26, pp. 250-255, 1986

[9] Chien, C.-H.: *Reconstruction and recognition of 3-D objects from occluding contours and silhouettes*, Ph.D. dissertation, Dep. Elec. Comput. Eng., University of Texas, Austin, 1987

[10] Chien, C.-H. and Aggarwal, J.K.: *Model Construction and Shape Recognition from Occluding Contours*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 11, No. 4, pp. 372-389, 1989

[11] Cipolla, R. and Blake, A.: *The Dynamic Analysis of Apparent Contours*, Proceedings Third International Conference on Computer Vision, Osaka, pp. 616-623, 1990

[12] Cipolla, R. and Blake, A.: *Surface Shape from the Deformation of Apparent Contours*, International Journal of Computer Vision, Vol. 9, No. 2, pp. 83-112, 1992

[13] Giblin, P. and Weiss, R.: *Reconstruction of Surfaces from Profiles*, Proceedings 1rst International Conference on Computer Vision, London, pp. 136-144, 1987

[14] Horaud, R. and Brady, M.: *On the geometric interpretation of image contours*, Artificial Intelligence, Vol. 37, pp. 333-353, 1988

[15] Jackins, C.L. and Tanimoto, S.L.: *Oct-trees and their use in representing three-dimensional objects*, Computer Graphics Image Processing, Vol. 14, pp. 249-270, 1980

[16] Jarvis, R.A.: *A perspective on range finding techniques for computer vision*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-5, No. 2, 1983

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

[18] Liedtke, C.-E., Busch, H. and Koch, R.: *Shape Adaption for Modelling of 3D Objects in Natural Scenes*, Proceedings IEEE Conference on Computer Vision and Pattern Recognition, pp. 704-705, 1991

[19] Martin, W.N. and Aggarwal, J.K.: *Volumetric Descriptions of Objects from Multiple Views*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-5, No. 2, pp. 358-366, 1983

[20] Meagher, D.: *Geometric modelling using octree encoding*, Computer Graphics Image Processing, Vol. 19, pp. 129-147, 1982

[21] Pavlidis, T.: *Algorithms for Graphics and Image Processing*, Computer Science Press, 1982

[22] Requicha, A.A.G.: *Representations for rigid solids: Theory, methods, and systems*, Comput. Surveys, Vol. 12, No. 4, pp. 437-463, 1980

[23] Thompson and Mundy: *3D model matching from an unconstrained viewpoint*, Proceedings IEEE Conference on Robotics and Automation, 1987

[24] Stenstrom, J.R. and Connolly, C.I.: *Constructing Object Models from Multiple Images*, International Journal of Computer Vision, Vol. 9, No. 3, pp. 185-212, 1992