Geometric Clipping Using Boolean Textures

William E. Lorensen
General Electric Company
Corporate Research and Development
Schenectady, New York 12345

Abstract

Texture mapping is normally used to convey geometric detail without adding geometric complexity. This paper introduces boolean textures, a texture mapping technique that uses implicit functions to generate texture maps and texture coordinates. These boolean textures perform clipping during a renderer's scan conversion step. Any implicit function is a candidate boolean texture clipper. The paper describes how to use quadrics as clippers. Applications from engineering and medicine illustrate the effectiveness of texture as a clipping tool.

Introduction

Computer graphics applications in medicine, engineering and science often produce images that contain multiple objects that occlude each other. To understand the complex spatial relationships that exist in these renderings, users look for graphical tools that can efficiently manipulate the content of the views. Numerous graphical techniques exist to help a user selectively view multiple objects or objects that have multiple surfaces. Standard techniques include: front and back clipping, transparency, selective object display, wireframe and color.

Texture mapping is a popular computer graphics technique that is usually used to add visual detail to a rendered image without adding geometric complexity. Texture mapping, once limited to expensive flight simulation hardware or special purpose rendering software, is now becoming commercially available in general purpose graphics hardware from companies like Silicon Graphics, Hewlett Packard and Evans and Sutherland and in software [20].

This paper introduces boolean textures, a new visualization technique that uses texture mapping to clip or color graphics primitives according to their distance from implicit surfaces. Since it uses texture mapping to achieve its effects, operating during the scan conversion process, it is a visualization technique, not a modeling technique.

After a survey of clipping algorithms and texture mapping, we demonstrate the technique first for single planar surfaces and then for multiple, higher order surfaces. We use these basic elements to create a boolean texture tool box that includes clippers, telescopes, and outliners. Medical and engineering applications illustrate the applicability of the approach to practical problems.

Related Work

Boolean textures cover two broad areas: clipping and texture mapping. A brief review of related work follows.

Clipping

Clipping can be performed during either modeling or rendering. During modeling, stringent topological requirements require robust, geometric clipping algorithms that produce new geometric entities. These geometric algorithms can be time consuming and complex [9,19]. Also, different primitives require different clipping algorithms. If other than planar clipping is desired, the algorithm complexity increases. However, new models are sometimes a requirement, justifying the expense and complexity of geometric clipping. But this paper addresses rendering, not modeling, and, geometric clipping algorithms are too expensive to be used during the rendering cycle.

Clipping performed during the rendering process must be efficient especially if the location and orientation of the clipping primitives depend upon the view. The Silicon Graphics GL library allows up to six planar clipping planes to be applied to geometric objects. The clipping is performed by dotting each vertex with the clipping planes that have been transformed into eye space. Vertices whose dot products are negative are clipped.

Several authors report clipping that uses the graphics Z-buffer. Feiner [7] uses z-buffer-based picking to determine which objects are occluded. Lucas [12] performs arbitrary convex shape clipping with an additional z-buffer. The clipping primitives are rendered into the two z-buffers,

1070-2385/93 $3.00 © 1993 IEEE

maintaining the front and back extremes of clipping volume. Rossignac [16] uses a parity checking scanline algorithm to identify and render internal surfaces.

Texture Mapping

Texture mapping was first introduced to the graphics community by Catmull [5]. Catmull used texture to modify color and intensity in geometric renderings. Blinn extended texture to model mirror reflections [2] and bumpy surfaces [3]. Shadow mapping [14] generates shadows from a depth map. Alpha [13], or opacity can also exist in a texture map. Regardless of the final intent, texture mapping is a two step process. First, a texture map of intensities, colors and / or opacities is created. This map is normally a 1D or 2D image with 8, 16, 24, or 32 bits per pixel. Then, each vertex of the geometric model is assigned a texture coordinate based on some mapping from 3D Euclidean space into 2D texture space. For example, for models that are basically spherical, a mapping from cartesian coordinates to latitude / longitude is a viable texture coordinate generator. During scan conversion, the graphics hardware or software interpolates the texture coordinates using the same scheme used for vertex colors and normals. For polygonal models, the interpolation is usually linear. The interpolated texture coordinate becomes an index into the texture map. The actual texture value can be interpolated from neighboring pixels using nearest–neighbor or bi–linear interpolation. Heckbert [8] surveys texture mapping and describes a number of important issues including texture coordinate generation and aliasing.

Boolean Textures

Texture mapping algorithms can be characterized by a model (or texture map) and an index (or texture coordinate).

- Geometric textures model reflectance of surfaces, indexed by geometry.
- A bump map models roughness, indexed by geometry.
- An environment map models mirror reflections, indexed by direction of reflection.
- A shadow map models light source obscurity, indexed by location of a point.

The boolean textures introduced in this paper model inside / outside relations, indexed by distance from a surface. The boolean texture map contains colors, intensities and opacities that vary with distance from the clipping surfaces.

First we describe planar clipping with one–dimensional boolean textures and later extend the technique to higher order surfaces and texture dimensions.

Texture Coordinate Generation

The implicit equation of a plane located at (a, b, c) with unit normal $\overline{N} = (n_x, n_y, n_z)$ is:

$$F(x, y, z) = n_x x + n_y y + n_z z - (an_x + bn_y + zn_z)$$

The plane partitions space onto two half–spaces, one on the side of the normal, the other on the opposite side. Evaluation of F(x, y, z) at any point produces a value < 0, $= 0$, or > 0. Points that lie *inside* the half–space created by the plane have values $F(x, y, z) < 0$; points on the planar surface, $F(x, y, z) = 0$; and for points *outside*, $F(x, y, z) > 0$. Also, for a plane, the value of $|F(x, y, z)|$ is the Euclidean distance of the point (x, y, z) from the plane and the sign of the result tells on which side of the plane the point resides. For each point in our geometric model, we calculate this signed distance, transform it into valid texture coordinate values, and assign the result as the texture coordinate for the point. Most systems use texture coordinates in the range (0, 1), so we offset the distance by .5 to move the point corresponding to zero distance to the center of this range. Any point inside the plane has a texture coordinate $< .5$, any outside has a coordinate $> .5$ and any point on the plane has texture coordinate $= .5$.

Texture Map Generation

Each point in the geometric model now has a texture coordinate that corresponds to its signed distance from a plane. This coordinate classifies a point as to whether it is inside, on, or outside the planar surface. Similarly, we design a texture map to partition texture space (0, 1) into three regions: pixels to the left of center control the color and opacity of points inside the plane, pixels at the center control color and opacity of points on the plane and pixels to the right affect the appearance of points outside the plane. Figure 1 shows a boolean intensity texture. This example sets intensity to a value less than 1 for geometric points inside the clipping surface and intensity to one for points outside the surface. With these matching texture coordinates and maps, the polygon scan conversion process interpolates the texture coordinate, uses it to look up a color, and assigns the result to the rendered pixel. If a transition from inside to outside occurs during the scan conversion, the rendered pixels change in appearance from inside pixels to outside pixels. Since we only need to know whether the point is in one of three finite states we can apply nearest neighbor interpolation to construct a pixel value from the texture map. In fact, we only need a three element texture map to accomplish the task! Further, if we include points that are on the surface with those that are inside, we only need a 2 element texture map.

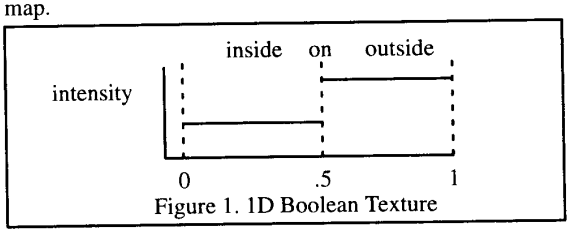

Figure 1. 1D Boolean Texture

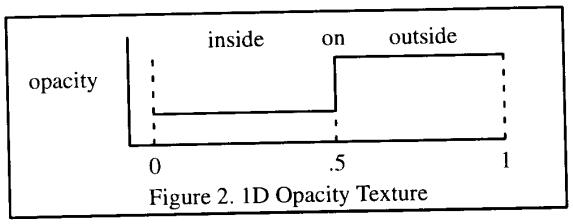

Figure 2. 1D Opacity Texture

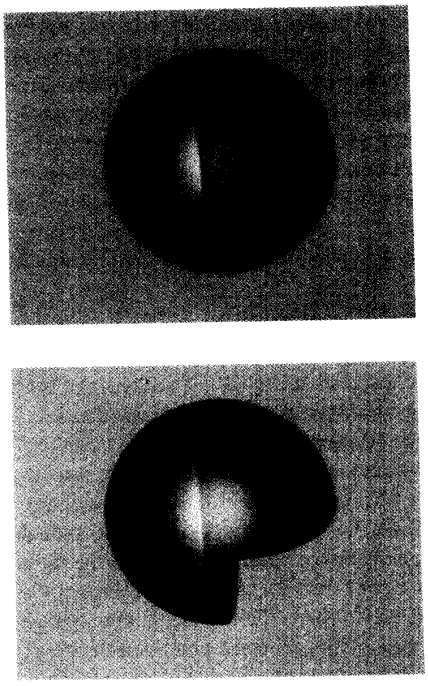

Figure 3. Boolean textures on a sphere.

Clipping with Opacity

The texture map in Figure 2 contains opacity. Points inside the plane can be made semi–transparent and other points opaque. Combinations of opacities and intensities for the three regions produce interesting results. Figure 3 shows the results of applying intensity and opacity boolean textures to a sphere. Here the clipping primitives are two orthogonal planes passing through the center of the sphere.

Extensions to Other Functions

We can apply the same procedure to any function that distinguishes between inside and outside. The quadrics are a powerful implicit family that have found applications in computer graphics. The quadrics encompass spheres, ellipsoids, cones, and cylinders. A homogeneous representation of the quadric is convenient to work with:

$$PQP^t = 0$$

where

$$Q = \begin{bmatrix} a & b & c & d \\ b & e & f & g \\ c & f & h & i \\ d & g & i & j \end{bmatrix}$$

and

$$P = [x \ y \ z \ 1]$$

Some useful quadrics include:

$Q = \begin{bmatrix} 0 & 0 & 0 & n_x \\ 0 & 0 & 0 & n_y \\ 0 & 0 & 0 & n_z \\ n_x & n_y & n_z & 0 \end{bmatrix}$, a plane with normal n_x, n_y, n_z passing through $(0, 0, 0)$.

$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -r^2 \end{bmatrix}$, a sphere of radius r, centered at $(0, 0, 0)$.

$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -r^2 \end{bmatrix}$, a cylinder of infinite extent with radius r, centered at $(0, 0, 0)$.

$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-r}{d} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$, a cone of infinite extent with apex at $(0, 0, 0)$ and radius r at a distance d from the apex.

Transformation of a quadric produces a quadric using the following relation: $Q' = M^{-1}QM^{-1^t}$ [4]. This transformation property makes modeling with quadrics straight forward. Specify the quadric in a convenient zero–centered coordinate system and translate and rotate it into the desired position.

Rockwood [15] suggests using the value of the function F(x, y, z) as a measure of distance. He calls this measure the *algebraic distance* of F(x, y, z). The *algebraic distance* has the same sign as the Euclidean distance but varies as F(x, y, z). Since the inside / outside crossing is the same, we can assign this readily calculated value as the texture coordinate. We use the same texture maps as we did for the planar case. The *algebraic distance* is acceptable for a nearest neighbor texture map interpolation scheme because nearest neighbor does not care how far away the point is from the inside / outside boundary. But, if we use higher order inter-

polation schemes for texture determination, we need a more meaningful measure of distance.

We can derive the distance between a point and a quadric by finding the intersection of a line with a quadric. From Blinn [1], define two points on the line and represent all points on the line as a linear combination of these points.

$$P = \alpha P_1 + \beta P_2$$

Then solve the quadratic equation

$$0 = PQP^t$$

$$0 = (\alpha P_1 + \beta P_2)Q(\alpha P_1 + \beta P_2)^t$$

$$0 = \alpha^2(P_1QP_1^t) + 2\alpha\beta(P_1QP_2^t) + \beta^2(P_2QP_2^t)$$

$$0 = A\alpha^2 + 2B\alpha\beta + C\beta^2$$

$$(\alpha_1,\beta_1) = (-B + \sqrt{B^2 - AC}, A)$$

$$(\alpha_2,\beta_2) = (-B - \sqrt{B^2 - AC}, A)$$

For this application, we select $P_1 = (x, y, z, 1)$ and $P_2 = P_1 + \overline{N}(x, y, z)$. $\overline{N}$ is the normal to the quadric at (x, y, z) and is:

$$N(x, y, z, 1) = 2(x, y, z, 1)\, Q \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

Then, the distance from P_1 along the normal $N(x, y, z)$ is $\min(\| P_1 - P(\alpha_1, \beta_1) \|, \| P_1 - P(\alpha_2, \beta_2) \|)$.

This calculation has about four times the expense of the $F(x, y, z)$ evaluation and should only be used if interpolation other than nearest neighbor is required.

Extensions to Multiple Surfaces

We can extend the boolean textures to multiple planes or quadrics. Texture coordinate assignment is made by taking the maximum $|F(x, y, z)|$ for all surfaces participating in the clip. This models the intersection of the surfaces. The minimum $|F(x, y, z)|$ models the union of the surfaces. In fact, any of the popular implicit modeling [10] techniques can be used to generate more complicated inside / outside relationships. However, as Rockwood [15] points out, blended surfaces require more sophisticated distance measures.

Extensions to Multiple Texture Dimensions

Two–dimensional texture maps provide the opportunity to perform boolean clipping with two surfaces. Here, texture coordinates are independently generated for two implicit surfaces. The relationship between the inside / outside functions of the two surfaces is readily described in the 2D texture map. Four regions define the combinations of inside and outside for two surfaces, Figure 4. The transitions between the four regions affect the appearance of edges between the surfaces. Figure 5 shows the sixteen possible labelings of the 2D boolean texture map. For example, case 2 clips all points that are outside surface A and inside surface B. Figure 6 shows the results of applying the sixteen possible combinations of two elliptical cylinders clipping a sphere. In all sixteen cases, once the texture coordinates are calculated, the different clipping schemes are achieved by changing the texture maps. The texture maps also darken the intensity as scan conversion travels from inside pixels to outside pixels.

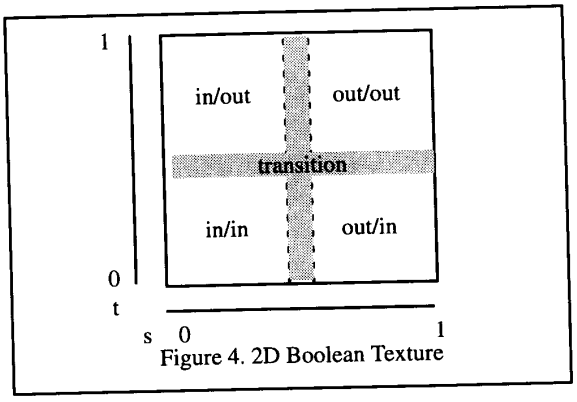

Figure 4. 2D Boolean Texture

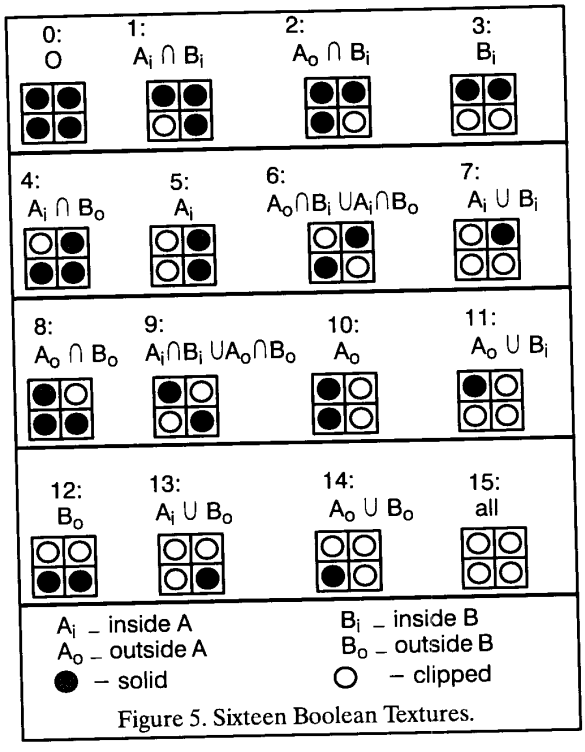

Figure 5. Sixteen Boolean Textures.

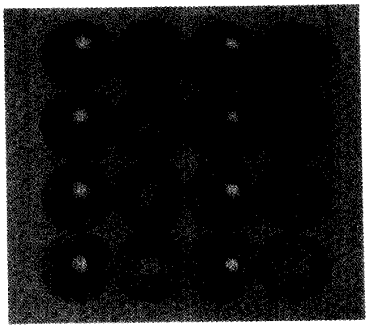

Figure 6. Two Cylindrical Clippers

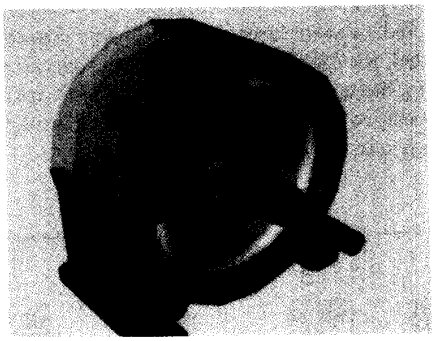

Figure 7. Octant cuts on a motor.

A Texture Clipping Tool Box

With an understanding of the basics of boolean textures, we proceed to develop a number of useful tools.

Selective Clipping

Selectively clipping of objects in a scene produces cutaway views that help show the spatial relations between multiple objects. Figure 7 shows a CAD model of a motor. Each part of the motor is clipped with two orthogonal clipping planes. Regions inside the intersection of these two planes are transparent. The transition between inside and outside is darkened to outline the intersections.

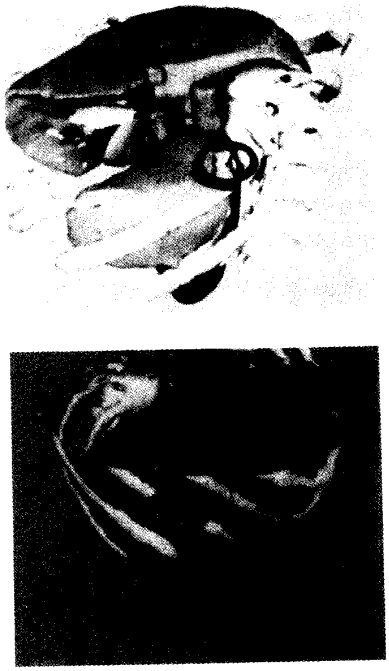

Figure 8. Telescope.

Telescope

The telescope uses boolean textures to keep an *eye* on an interesting feature in a model as the camera moves about the scene. A clipper created from two quadrics, a cylinder and a plane, is placed at the point of interest, pointed towards the camera. As the camera location changes, we transform the quadric, recalculate the texture coordinates and render the new view. We use a texture map that shows all pixels that are outside the cylinder and outside the plane. This clips away all pixels that are inside the cylinder and are not outside the plane. Figure 8 shows a surgical planning application using the telescope tool. As the view changes, the telescope remains pointed at a tumor on the kidney.

(See color plates, p. CP-28.)

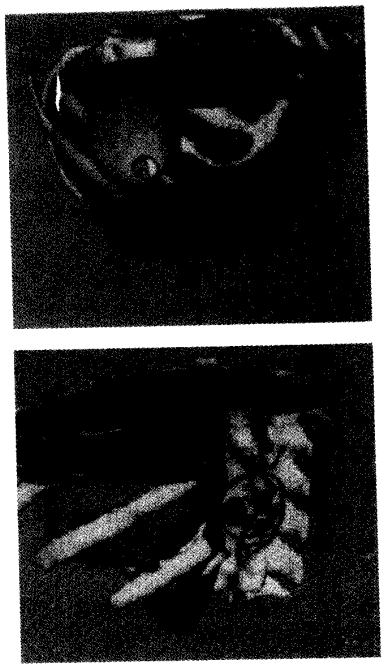

Figure 9. Simulated Ultrasound Surgery.

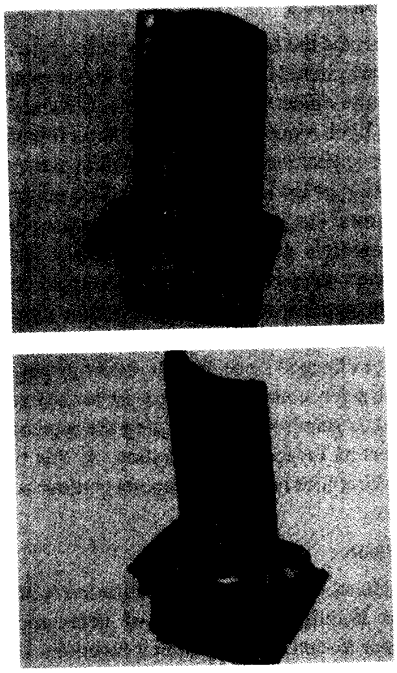

Figure 10. Solid Cuts.

Simulated Surgery

Figure 9 shows two views of an ultrasound surgery simulation [6]. A conical quadric focuses at a tumor on the kidney. A geometric model of a cone is rendered in the clipped region. The darkened intersections of the cone and anatomy clearly indicate the regions affected by the ultrasound heat source.

Pseudo–Solid Clipping

Clipping often reveals the back side of interior surfaces. Such renderings can be confusing. We find it useful to shade the back sides of surfaces with different colors by changing the material properties that we use when the surface normal points away from the viewer.

The turbine blade in Figure 10 has been clipped with 60 clipping planes. The original data was obtained using an industrial CT scanner. Over 1.5 million triangles were created from the 512 by 512 by 300 slices using Marching Cubes [11]. Triangle decimation [17] reduced the triangle count to under 100,000. The resulting blade model was clipped using two boolean textures: one revealing triangles within the intersection of the clipping planes, the other showing triangles outside the planes. Surfaces that border air are metal colored while those bordering metal are colored red. Both renderings in Figure 10 use the same texture coordinates for vertices. Only the texture map changes for the two views.

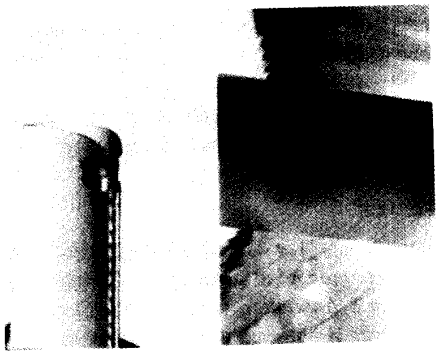

Figure 11. Overview of an Internal Flythrough.

Where Am I?

Exploration of complex models often requires a close up view of internal geometry. Wandering through the zoomed in features of these models, the viewer can lose track of location with respect to the original model. Figure 11 solves this problem with two views of the model: a close up and an overview. The images on the left use a boolean texture with an elliptical cylinder to clip a hole in the surface of the turbine blade. A planar quadric limits the extent of the cylindrical clipper. The sphere within the hole shows the camera location of the close up view on the right. As the location of the close up camera changes, texture coordinates of the overview model are regenerated.

(See color plates, p. CP-28.)

Implementation

We implemented the boolean textures in an object–oriented visualization system [18]. The boolean texture map is a procedural texture object that generates the textures described in the paper upon request. Instance variables for map resolution, region appearance, and interpolation scheme permit ready variation of the boolean texture. Two texture coordinate generators, *cut_planes*, and *cut_quadrics* permit clipping with multiple planes and two quadrics. Our object–oriented system keeps track of the last modified time of any object in an application. The visualization pipeline uses these times and only re–evaluates those portions of the pipeline that have changed since the last render operation. If the clipping surface remains unchanged, no texture coordinate computation is required. So, changing the type of boolean clip becomes an inexpensive operation. At this time, only our Silicon Graphics hardware supports textures with opacity.

Conclusions

Boolean textures offer a simple visualization technique for clipping or highlighting portions of geometric models based on their distances from implicit functions. Any function that can distinguish between inside and outside and has a distance metric is a candidate clipper. Texture clipping is not a substitute for geometric clipping, but rather provides an efficient alternative when visualization is the primary objective.

We expect that as texture mapping becomes more available, boolean textures will be the basis for new tools in many visualization systems.

References

[1] Blinn, J. F., "The Algebraic Properties of Homogeneous Second Order Surfaces," in *Course Notes: Modelling and Animating with Implicit Surfaces*, ed. B. Wyvill, J. Bloomenthal, Association for Computing Machinery, August 1990.

[2] Blinn, J. F. and Newell, M. E., "Texture and Reflection in Computer Generated Images," *Communications of the ACM*, vol. 19, no. 10, pp. 542–547, October 1976.

[3] Blinn, J. F., "Simulation of Wrinkled Surfaces," *Computer Graphics*, vol. 12, no. 3, pp. 268–292, August 1978.

[4] Blinn, J. F., "A Generalization of Algebraic Surface Drawing," *ACM Transactions on Graphics*, vol. 1, no. 3, pp. 235–256, July 1982.

[5] Catmull, E., "A Subdivision Algorithm for Computer Display of Curved Surfaces," PhD Thesis, University of Utah, Salt Lake City, UT, 1974.

[6] Cline, H. E., Schenck, J. F., Hynynen, K., Watkins, R. D., Souza, S. P., and Jolesz, F. A., "MR–Guided Focused Ultrasound Surgery," *Journal of Computer Assisted Tomography*, vol. 16, no. 6, pp. 956–965, 1992.

[7] Feiner, S. K. and Seligmann, D. D., "Dynamic 3D Illustrations with Visibility Constraints," in *Scientific Visualization of Physical Phenomena*, pp. 525–543, SpringerVerlag, June 1991.

[8] Heckbert, P. S., "Survey of Texture Mapping," *IEEE Computer Graphics and Applications*, vol. 6, no. 11, pp. 56– 67, November 1986.

[9] Hubl, J. and Herman, I., "Modelling Clip: Some More Results," Computer Graphics Forum, vol. 9, no. 2, pp.101–107, 1990.

[10] Kleck, J., "A Starter Toolkit for Modeling with Implicit Surfaces," in *Course Notes: Modelling and Animating with Implicit Surfaces*, ed. B. Wyvill, J. Bloomenthal, Association for Computing Machinery, August 1990.

[11] Lorensen, W. E. and Cline, H. E., "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *Computer Graphics*, vol. 21, no. 3, pp. 163–169, July 1987.

[12] Lucas, B., "A Scientific Visualization Renderer," in *Proceedings of Visualization '92*, pp. 227–234, October 1992.

[13] Porter, T. and Duff, T., "Compositing Digital Images," *Computer Graphics*, vol. 18, no. 3, pp. 253–259, July 1984.

[14] Reeves, W. T., Salesin, D. H., and Cook, R. L., "Rendering Antialiased Shadows with Depth Maps," *Computer Graphics*, vol. 21, no. 4, pp. 283–291, July 1987.

[15] Rockwood, A. P., "The Displacement Method for Implicit Blending Surfaces in Solid Models," *ACM Transactions on Graphics*, vol. 8, no. 4, pp. 279–297, October 1989.

[16] Rossignac, J., Megahed, A., and Schneider, B., "Interactive Inspection of Solids: Cross–sections and Interferences," *Computer Graphics*, vol. 26, no. 2, pp. 353–360, July 1992.

[17] Schroeder, W., Zarge, J., and Lorensen, W., "Decimation of Triangle Meshes," *Computer Graphics*, vol. 26, no. 2, pp. 65–70, July 1992.

[18] Schroeder, W., Lorensen, W., Montanaro, G., and Volpe, C., "Visage: An Object–Oriented Scientific Visualization System," in *Proceedings of Visualization '92*, pp. 219–226, October 1992.

[19] Sutherland, I. E. and Hodgman, G. W., "Reentrant Polygon Clipping," *Comm. of the ACM*, vol. 17, no. 1, pp. 32–42, January 1974.

[20] Upstill, S., in *The RenderMan Companion*, Addison–Wesley, Reading, MA, December 1989. ISBN: 0–201–50868–0.

Geometric Clipping Using Boolean Textures, W.E. Lorensen, pp. 268-274.

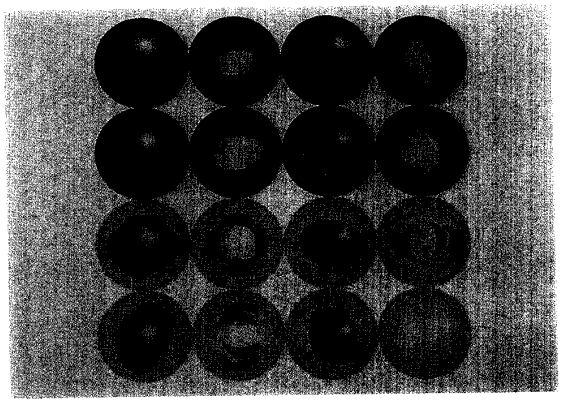

Figure 6: Two cylindrical clippers.

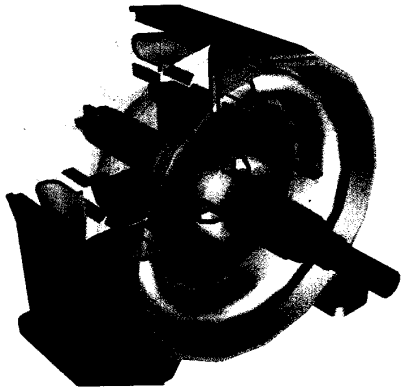

Figure 7: Two octant cuts on a motor.

Figure 8: A telescope focused on a tumor.

Figure 9: Simulated ultrasound surgery.

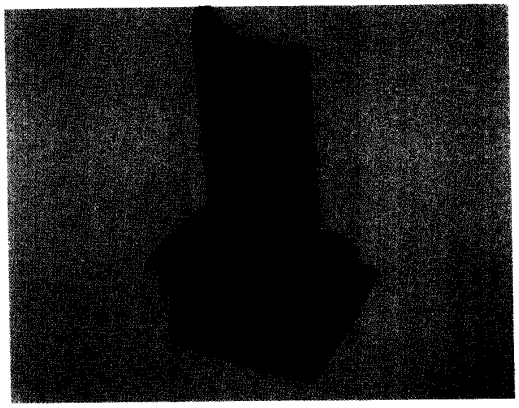

Figure 10: 60 clipping planes.

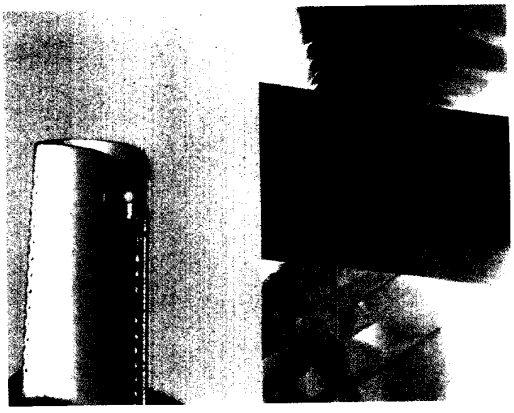

Figure 11: Turbine blade flythrough.