

Interior/Exterior Classification of Polygonal Models

F.S. Nooruddin and Greg Turk

GVU Center, College of Computing, Georgia Institute of Technology

Abstract

We present an algorithm for automatically classifying the interior and exterior parts of a polygonal model. The need for visualizing the interiors of objects frequently arises in medical visualization and CAD modeling. The goal of such visualizations is to display the model in a way that the human observer can easily understand the relationship between the different parts of the surface. While there exist excellent methods for visualizing surfaces that are inside one another (nested surfaces), the determination of which parts of the surface are interior is currently done manually.

Our automatic method for interior classification takes a sampling approach using a collection of direction vectors. Polygons are said to be interior to the model if they are not visible in any of these viewing directions from a point outside the model. Once we have identified polygons as being inside or outside the model, these can be textured or have different opacities applied to them so that the whole model can be rendered in a more comprehensible manner. An additional consideration for some models is that they may have holes or tunnels running through them that are connected to the exterior surface. Although an external observer can see into these holes, it is often desirable to mark the walls of such tunnels as being part of the interior of a model. In order to allow this modified classification of the interior, we use morphological operators to close all the holes of the model. An input model is used together with its closed version to provide a better classification of the portions of the original model.

Keywords: Visibility, Surface Classification, Rendering, Interior Surfaces

1 Introduction

In this paper we present a method for determining the interior and the exterior portions of a given polygonal model. Our motivation is that for many visualization applications it is desirable to display surfaces in such a way that a human observer can clearly see the relationships between different parts of the object. While excellent techniques exist for displaying nested surfaces [4, 5, 6, 9], the determination of which surfaces are exterior to the model and which are

interior is typically not an automated process. Usually this classification is done either by hand or by connected component analysis. For many models in application areas such as medicine or industrial design, connected component analysis is not helpful because different parts of the model may be connected to each other by holes, tubes or thin structures. Our method overcomes this limitation of connected component analysis.

For some applications, the nature of the data can give clues as to whether a portion of a surface should be considered interior. For example, with medical data from CT or MRI, the volume densities may be used to help classify different tissue types. Unfortunately, such approaches fail when a user wishes to visualize the interior structure of an organ that is relatively uniform in tissue type such as the chambers of the heart, the alveoli of the lungs or the folds and interior cavities of the brain. Likewise, some CAD data may be tagged with different part identifiers or surface properties. In some cases, however, these tags have been lost or the model that is being visualized is from actual captured data for parts inspection such as CT. In these cases, once again we require an automated method of identifying interior parts without the aid of pre-specified tags.

In this paper we present a new technique that classifies each polygon of a given model as being either interior or exterior to the surface. We can then use different rendering styles to show off these different portions of a surface. Our method uses a collection of viewing directions in order to classify each point as being a part of the exterior or the interior of the model. When processing models with holes, we first use volumetric morphology to obtain a closed version of the input model. The closed model is used in conjunction with the original model to provide a better classification for the surfaces of the original model.

The remainder of the paper is organized as follows. In Section 2 we review existing methods for viewing interiors of models. In Section 3 we present our definition of visibility. In Section 4 we describe how to generate a “deep” depth buffer for each viewpoint. In Section 5 we describe how we use the depth buffers to classify a polygon as being exterior or interior to the model. In Section 6 we describe the use of morphology to close holes in models to get improved results. In Section 7 we present the results of our method. Section 8 provides a summary and describes possible future work.

2 Previous Work

There have been several techniques in graphics and visualization that are tailored towards viewing the interior of surfaces, including wireframe drawing, clipping and volume rendering. We will briefly review each of these below.

Wireframe drawing of polygonal models has been used in computer graphics at least since the 1950's. Until fast polygon rasterization hardware came along in the 1980's, wireframe drawings on raster or calligraphic displays was the viewing method of choice for rapid examination of 3D ob-

e-mail: {fnsn,turk}@cc.gatech.edu

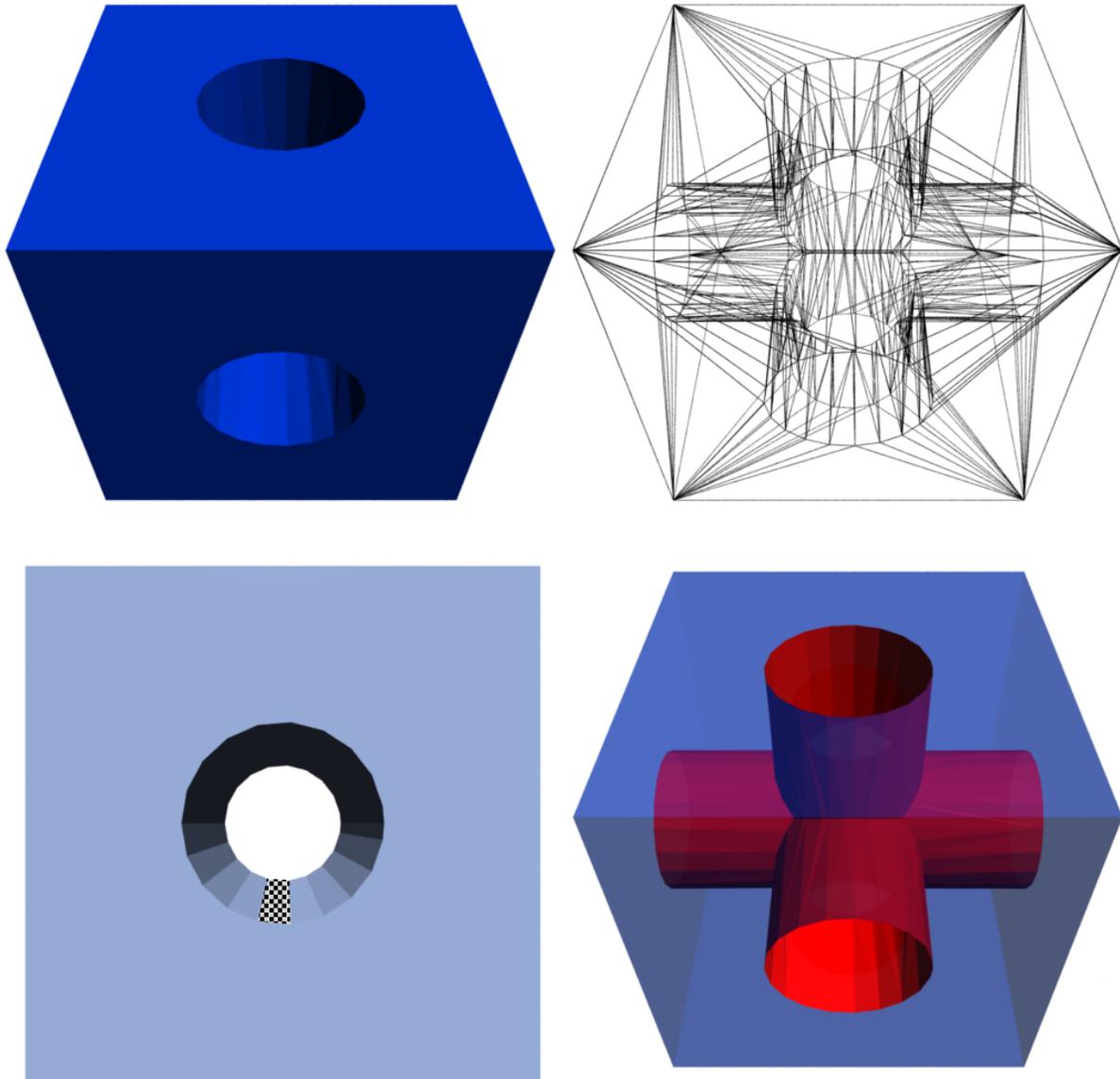


Figure 1: Top left: Cube with three holes, rendered as an opaque surface. Top right: wireframe rendering of the same cube. Bottom left: Another view of opaque cube with an internal polygon highlighted. Bottom right: Cube with a translucent external surface and an opaque interior surface.

jects. At present, wireframe drawings of models are sometimes used to show some of the interior detail of simple models. Unfortunately there is considerable ambiguity in viewing such a wireframe model due to lack of occlusion between lines at different depths. Moreover, polygon size greatly affects the understanding of such images. An inner surface that is tiled with just a few large polygons may be difficult to see because the surface is represented only by a small number of lines. Figure 1 (top right) shows a wireframe drawing of a cube with holes.

Clipping planes and solids are often used in visualization to allow an observer to see into a portion of a model. Rossignac et al. demonstrated a variety of solid clipping techniques for CAD model inspection, including methods for cross-hatching those portions of a model that have been

sliced [10]. Cutting planes, half-planes and other shapes have been used in volume rendering to see into the interiors of objects. Medical data has been the application area where this has been put to use the most extensively [12]. Unfortunately, creating clipping planes or clipping volumes that are tailored for a given object is a user-intensive task.

Volume rendering techniques allow some portions of a model to be made translucent or entirely transparent, allowing an observer to see interior parts of a model [2, 6]. Segmentation of a volume model, whether it is done automatically or by hand, allows regions to be tagged as translucent or opaque for better comprehension of interior surfaces. Automatic segmentation is very useful if the model to be visualized is composed of different materials with differing voxel densities, as is common with medical data. In many cases,

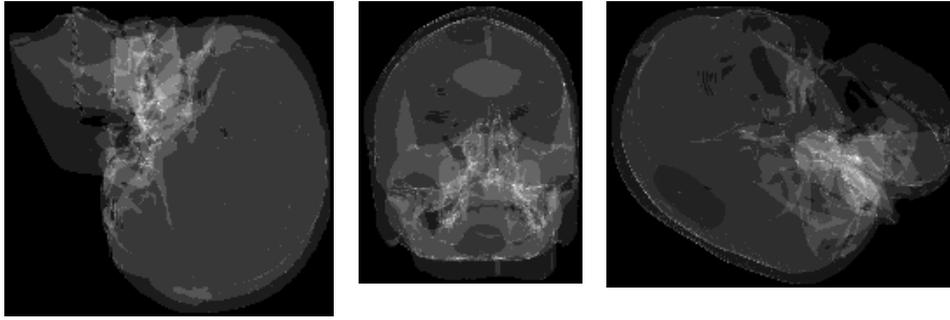


Figure 2: Gray-scale representation of three of the depth buffers created for a skull dataset. Brighter pixels are locations where more depth samples are present in the buffer.

however, such as the turbine blade or the skull of Figure 6, there is no way to separate out the interior structure based on density values.

There are several published methods for viewing multiple nested surfaces in order to show the context of one surface inside another. These methods render interior surfaces as opaque, but place a pattern on a translucent exterior surface in order to better understand the shape of this partially transparent surface. Interrante has used both strokes and line integral convolution to decorate translucent exterior surfaces [4, 5]. Rheingans used re-triangulation of a given polygonal model in order to create regular patterns such as disks on a partially transparent exterior surface [9]. Given an automatic technique for identifying interior and exterior portions of a surface, any of these techniques could be put to better use to show the relationships between these different portions of a surface.

3 Defining Visibility

Our first task is to define what we mean for a portion of a surface S to be exterior or interior. These definitions will depend on the visibility of the part of the surface in question. Intuitively, we will say that a point \mathbf{p} is on the exterior of a surface if there is some sufficiently distant viewpoint \mathbf{q} from which \mathbf{p} is visible. We say “sufficiently distant viewpoint” because there is always some position \mathbf{q} from which a given point will be visible, and we wish to disallow those viewpoints that are too close (or perhaps inside) the surface.

To define exterior, we will make use of the concept of the *extremal points* of a surface along a line. A line L that passes through two points \mathbf{p}_1 and \mathbf{p}_2 can be defined as the set of points $L = \{\mathbf{p}_1 + t(\mathbf{p}_2 - \mathbf{p}_1) : t \in \mathbb{R}\}$. Consider the set of points of the surface along a line, $T = S \cap L$. Any point \mathbf{p} in this set can be written as $\mathbf{p} = \mathbf{p}_1 + t(\mathbf{p}_2 - \mathbf{p}_1)$ for some t , which can allow us to define a function $t(\mathbf{p}) = (\mathbf{p} - \mathbf{p}_1) / (\mathbf{p}_2 - \mathbf{p}_1)$. We will say that a point \mathbf{p} in T is an *extremal point* if the value of $t(\mathbf{p})$ takes on either the minimum or maximum value for all points in T . It is now easy to define interior and exterior. We define a point \mathbf{p} on S to be *exterior* if it is an extremal point for some line L . We will say \mathbf{p} on S is *interior* if it is not an extremal point for any line L .

It is a compute-intensive task to determine whether there exists a line for which a given point is extremal. For our own work, we have found it sufficient to consider a collection of lines that are parallel to one of a small set of directions V . The elements of V are vectors $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n\}$ that are evenly distributed over a hemisphere. We then consider

whether a given point \mathbf{p} is extremal along any of the lines $L = \mathbf{p} + t\mathbf{v}$ for $\mathbf{v} \in V$. In our method, we classify a point \mathbf{p} on a surface S as *exterior* if it is an extremal point along a line that passes through \mathbf{p} along at least one of the directions in V . We will classify a point as *interior* if it is not exterior. In the following sections we will describe a fast algorithm for performing this classification task.

4 Generating Depth Buffers

Given a set of viewing vectors $V = \{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n\}$, we seek a way to mark which polygons are visible along each viewing direction \mathbf{v}_i . A polygon p is classified as being inside the model if all the rays cast along \mathbf{v}_i strike another surface before hitting p . In order to do this process efficiently for large polygonal models, we use polygon scan-conversion with an orthographic projection to create “deep” depth buffers. Our “deep” depth buffers are similar in structure to the Layered Depth Images presented in [11]. In that work, each pixel in a LDI kept a list of depth samples representing depth samples further and further along a single line of sight. Similarly in our case, a depth buffer is a 2D array of pixels. Each “deep” pixel in the depth buffer contains not only the depth sample for the nearest polygon, but a sorted list of depth samples. Each depth sample in the list represents the intersection of a ray shot through the pixel with a polygon of the model. By using an orthographic projection, we are effectively sampling the model along a large collection of rays that are parallel to a given direction \mathbf{v}_i .

5 Classification

5.1 Using Depth Buffers

Once a depth buffer has been generated for each view, we are ready to classify all the polygons of the input model. To this end, we take a polygon p from the input model, and check how many views from which it is visible. For each view \mathbf{v}_k we select the depth buffer d_k . For each vertex of p , we find the depth pixel $d_k(i, j)$ onto which the vertex projects. The vertex is marked exterior if it is either the first or the last surface that the ray passing through $d_k(i, j)$ intersects. This corresponds to testing whether the vertex generated the first or last depth sample in the sorted list of depth samples stored at $d_k(i, j)$. The closest intersection to the vertex of p is reported back as the depth sample generated by that vertex. The polygon p is marked visible if all of its vertices are visible, and it is marked invisible otherwise.

We keep track of the number of depth buffers that declare p to be on the exterior and the number that classify p as being on the interior of the model. Figure 2 shows three out of the 40 depth buffers we used to classify the polygons of a skull model. When assigning the final classification to p , we require that at least m buffers classify it as being visible for the algorithm to declare that p is visible. This threshold value m is a user defined parameter in our implementation.

The reason a voting scheme is necessary to make the final visibility classification is that often all the depth buffers will not agree on the classification of a polygon. This can happen in several cases. The most common case is that of the input model containing tunnels. The polygons along the wall of a tunnel will be visible from a viewpoint that looks down the tunnel. This case is shown in Figure 1 (bottom left) where the marked polygon is visible from the current viewpoint, although it is embedded deep in the interior of the model. In this case, viewing directions orthogonal to the current viewpoint will vote that the highlighted polygon is interior to the model.

The other case where the depth buffers disagree on whether a polygon is visible or not is that of complex surfaces. Such models have polygons on the exterior surface that lie in highly occluded parts of the model. A large number of viewpoints must be used to adequately sample all the parts of the exterior surface. Polygons that lie in occluded parts of the model are only visible from a small number of viewing directions. Therefore we should expect that most of the depth buffers will vote for classifying such polygons as being invisible, while a small number of depth buffers will vote to make these polygons visible.

The classification threshold represents a trade-off between the two cases outlined above. If set too low, most polygons that lie on the interior of tubes embedded in the model will incorrectly be marked visible. However, polygons that lie on the exterior surface but in highly occluded parts of the model will be correctly marked visible. If it is set too high, then the algorithm will misclassify highly occluded polygons on the exterior surface of the model as being invisible. Polygons inside tubes will be marked invisible as expected. To alleviate this difficulty in choosing the correct classification threshold, we use morphology to close holes and collapse tunnels in complex models. This preprocessing step allows us to use the classification threshold solely to classify highly occluded polygons on surface the of a complex surface.

It is also possible to use the depth buffers to assign opacity values to the polygons of the model. These opacity values can then be used to render the model. For example, if 40 viewing directions are used to do the classification, then polygons visible from all 40 directions can be made almost transparent. Polygons that were not visible from any directions can be marked opaque. The opacity of polygons visible from a certain number of viewing directions can have an opacity value assigned to them based on the number of directions from which they are visible. The Skull and Motor models in Figure 6 were rendered using this scheme.

5.2 Processing Large Triangles

As described in Section 5.1, we project each of the vertices of a polygon p into all the depth buffers to determine the visibility of p . If p is a large polygon, then there is a good chance that the visibility will vary across p . Figure 1 (bottom left) shows this case. The highlighted polygon runs all the way across the tunnel. The solution that we employ to deal with

this problem is to break up large triangles into smaller ones. The decision of whether to subdivide a triangle is based on an edge-length tolerance that can be specified by the user. The smaller the edge-length tolerance, the greater the number of triangles that will be generated. And as smaller triangles result in better classification, the final result will benefit from a small edge-length threshold. In practise, the edge-length threshold is constrained by the largest file size on the machine being used, or by memory considerations.

We iterate over all the triangles in the input model, subdividing them until the largest edge in the model is below the user specified threshold. Once a triangle t has been subdivided into a n smaller triangles $\{t_0, t_1, \dots, t_n\}$, we process each of newly created triangles in the same way that a triangle belonging to the original model would be treated. For each t_i , we project its three vertices into all the depth buffers. Once all the depth buffers have voted on the visibility of t_i , we use the threshold to decide on the final classification of t_i .

6 Morphology

We use volumetric morphology to close holes and tunnels in complex models. Because our input models are polygonal, we require a robust method of converting a polygonal model to a volumetric representation. The technique that we use to voxelize the input polygonal model is described in [8]. To perform volumetric morphology, we use a 3D version of Danielsson's Euclidean 2D distance map algorithm [1]. The input to the distance map algorithm is a binary volume. Such a volume has voxels with the value 0 if they are not on the surface of the object, and 1 otherwise. The output of the distance map algorithm is a volume with the same dimensions as the input volume. This distance map volume contains, for each voxel, the distance to the nearest voxel on the surface. As expected, the distance map value of surface voxels is zero.

Given the distance map and a closing distance d , we can perform a volumetric *closing* of the volume. A closing operation consists of a *dilation* followed by an *erosion*. A dilation will convert all the background voxels within a distance d of the surface into surface voxels. The erosion operator converts all the surface voxels within distance d of any background voxel into background voxels. A closing operation will close holes and eliminate thin tunnels. Figure 3 shows the underside of both the original and a morphologically closed version of a turbine blade. The size of tunnels and holes in the volumetric description depends on the voxel resolution. In our work, we have found that the model should be voxelized at approximately 10 times the closing distance.

After performing a closing on the volumetric description of the model, we apply the Marching Cubes algorithm [7] to the volume to obtain an isosurface. This isosurface, which is a closed version of the input model, is used to generate another set of depth buffers from all the viewpoints. The depth buffers from the original and closed models are used together to classify all the polygons in the original model.

When classifying a polygon p in this case, we project p into both the original depth buffers and the closed depth buffers. We first check to see if p was near the exterior surface in the original depth buffer. As before, we project p into the depth buffer being processed. If p is too far away from both the minimum and maximum depth values, then it is marked as interior to the surface and the next depth buffer is considered. The maximum distance that p is allowed to



Figure 3: View of the four large holes in the base of the turbine blade (top) and the morphologically closed holes of the same model (bottom).

deviate from the minimum and maximum depth values is a user controlled parameter in our implementation. If both a morphologically closed and the original model are used to determine the visibility of the polygons, then this distance is equal to the closing distance used to produce the closed version of the model. If morphology is not used, then this distance should be set to the depth buffer resolution.

When p is determined to lie close to the exterior surface in the original model, we need to ensure that it is not part of a tunnel in the original model. To do this, we project p into the closed depth buffer and examine the list of depth values at the depth pixel that p projects to. If p was inside a tunnel in the original model, then it will not be close to any depth sample values in the closed model's depth buffers. This is due to the fact that the closed model does not have any tunnels, and therefore there will be no depth samples in its depth buffers at locations where the tunnel existed in the

original model.

7 Results

In this section we describe the results of applying our visibility classification to several polygonal models. All the images of Figure 6 are reproduced in the colorplate. For all of the results in this paper we used 40 viewing directions to create 40 depth buffers. These 40 directions were taken from the face normals of an icosahedron whose triangles were subdivided into four smaller triangles and whose vertices were then projected onto the surface of a sphere. Only face normals from half this model were used for the viewing directions since opposite faces define the same lines.

Figure 1 shows a model of a cube that has three holes punched through it. Rendering the surface as opaque does not show details of the geometry of how the holes meet in the interior of the model. The lower right of this figure shows a rendering of this cube based on our algorithm's classification of interior and exterior portions of the model. The exterior surface has been rendered partially transparent, and the interior is opaque. In this image, the interior structure is clearly evident. For this relatively simple model it would have been possible to come up with a mathematical expression to classify the interior and exterior. The other models we discuss below are far too complex to create any classification using such a mathematical expression.

Figure 4 (left) shows an opaque version of a turbine blade model that was created from CT data of an actual blade. The right portion of this figure shows only those polygons that have been classified as interior by our algorithm. Figure 6 (top row) shows more views of the same turbine blade model. These images were created using classification based on 40 viewing directions and the morphologically closed version of the object that is shown in the bottom of Figure 3. The detailed interior structure of the turbine blade is clearly revealed in these images. There are four large holes in its base that merge pairwise into two passageways. Each passageway snakes up, down, up, and then each meets up with many tiny holes on the left and right sides of the blade. In addition, the top right image shows a small tube (in the lower right of this image) that extends from the back to the side of the blade. This tube was previously unknown to us, even though we have used this turbine blade as a test model for several years, often rotating it interactively on a high-end graphics workstation. This indicates to us that images based on interior/exterior classification show promise for exploratory visualization of data.

Figure 6 (bottom row) shows a car engine model that was created by hand. This model has interesting topology in the sense that there are a number of parts connected by tubes and thin structures. In addition, this model has a large number of degeneracies such as T-joints, zero-area faces and non-manifold vertices. In this case, we did not perform a binary interior/exterior classification of the polygons of the motor model. Instead each polygon was assigned an opacity value based on the number of buffers that it was visible from. Again, 40 views and a morphologically closed version of the model were used to perform the opacity assignment. Polygons that were seen from more than 60% of the depth buffers were made almost transparent. And polygons that were visible from less than 10% of the buffers were made opaque. Other polygons were assigned opacities that varied linearly between these two limits. The closeup of the front part of the model shows quite clearly the gears and thin pipes in the

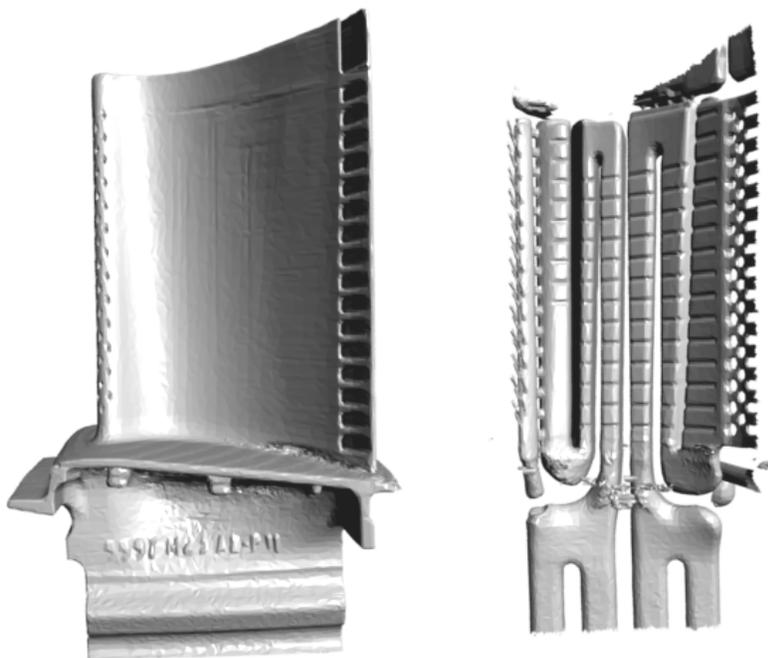


Figure 4: Opaque rendering of turbine blade (left), and just the interior surfaces (right) as classified by the method of this paper.

interior of the motor. The side view of the motor shows the cam-shaft of the engine in the lower part of the model.

Figure 6 (middle row) shows a dataset created from 124 CT slices of a human skull. Figure 5 shows two opaque renderings of the skull. 40 views and a morphologically closed version of the model were used to perform the opacity assignment. Again we did not do a binary interior/exterior classification of the polygons of the skull model, but instead assigned them opacities based on the number of directions from which they were visible. Turning the exterior surface transparent clearly reveals the two large sinus cavities on either side of the nose. Other detail is evident in these images, including a thin hollow tube on the forehead near the location where the plates of the skull join in the first two years after birth, and another small tube at the base of the jaw. The rightmost image of the back of the skull shows the *interior* suture lines, which are the interior versions of the more familiar suture lines on top of the skull that are visible from the outside.

Table 1 list the timing results of the different portions of the classification algorithm. In all cases, classification is performed in just a few minutes. Because our algorithm is proposed as a preprocessing step to a visualization method, the classification only needs to be done once for a given polygonal model.

8 Summary and Future Work

We believe that this paper makes two contributions to the field of visualization. First, we have identified a new problem, namely the problem of classifying exterior and interior portions of a polygonal model. The solution to this problem has potential applications in a number of areas in visualization. Second, we have proposed a specific solution to this problem – creating depth buffers of a model in several directions and classifying polygons based on whether they are

occluded in each of these directions. We further enhanced this method by closing holes using a morphological operator. We use classification information to make interior surfaces opaque and exterior surfaces partially transparent in order to visualize the relation between these portions of a model. This new approach gave results that revealed several features of the models that we did not know about prior to running the interior/exterior classification.

As with most first attempts to solve a new computational problem, the approach that we describe here is not ideal in all cases. High frequency variations in the surface can cause some small portions of a surface to be classified as interior even though a human observer would label them as exterior. Perhaps surface smoothing or some form of neighborhood information could be used to eliminate this problem.

Although our current implementation is quite fast, it may be possible to accelerate the method using hardware rendering rather than software scan-conversion to create the depth buffers. Finally, the results of our classification should be used in conjunction with a method that adds texture to exterior surfaces to better understand their shape.

9 Acknowledgements

We are grateful to Will Schroeder, Ken Martin and Bill Lorensen for the use of the turbine blade data that is included in the CD-ROM of their book *The Visualization Toolkit*. We thank Bruce Teeter and Bill Lorensen of General Electric and Terry Yoo of the National Library of Medicine for the skull data.

References

- [1] DANIELSSON, P. Euclidean Distance Mapping. *Computer Graphics and Image Processing*, 14, 1980, pp. 227-248.

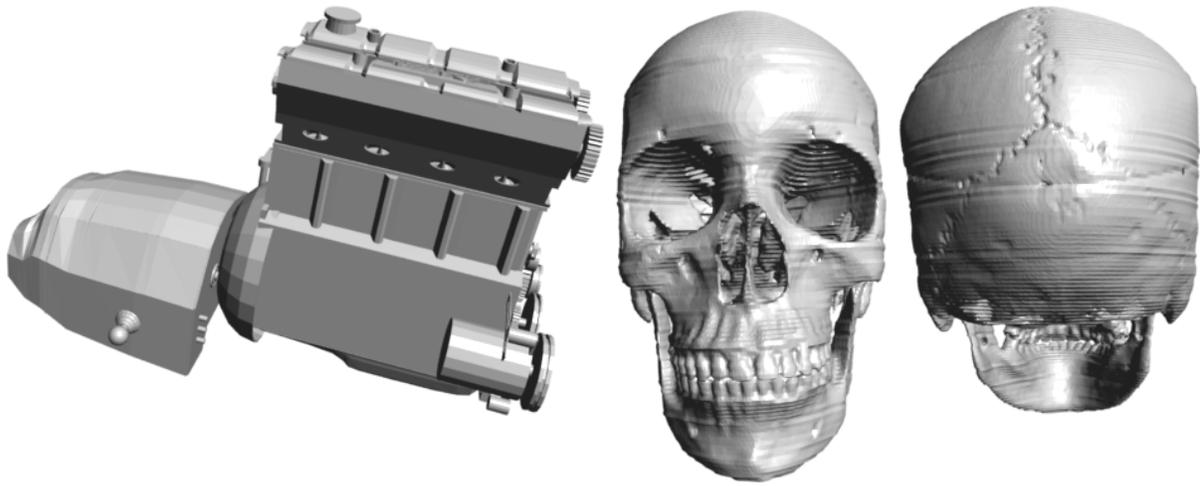


Figure 5: Opaque renderings of the Motor and Skull models. Notice the exterior suture lines on the back of the skull.

Time to Process Models (minutes:seconds)						
	Model Size (# polygons)	Triangles Processed (Small Triangles Generated)	Depth Buffer Creation	Morphology	Classification	Total
Blade	50,000	285,070 (375,994)	1:29	1:09	8:44	11:22
Skull	200,000	216,625 (29,371)	6:00	0:53	6:01	9:51
Motor	140,113	219,698 (122,219)	2:57	0:52	6:09	9:59
Cube	578	1,730 (1,820)	0:27	*	0:42	1:09

Table 1: This table shows the timing information for each stage of the classification process. All the timing measurements were taken on a SGI Octane with 256 Mb of Ram.

- [2] DREBIN, ROBERT A., LOREN CARPENTER, and PAT HANRAHAN Volume Rendering, *Computer Graphics*, Vol. 22, No. 4 (SIGGRAPH 88), August 1988, pp. 65-74.
- [3] EL-SANA, J. and A. VARSHNEY Controlled Simplification of Genus for Polygonal Models, *Proceedings of IEEE Visualization '97*, Oct. 19 - 24, Phoenix, Arizona, pp. 403-412.
- [4] INTERRANTE, VICTORIA, HENRY FUCHS and STEPHEN PIZER, Enhancing Transparent Skin Surfaces with Ridge and Valley Lines, *Proceedings of IEEE Visualization '95*, October 29 - November 3, Atlanta, Georgia, pp. 52-59.
- [5] INTERRANTE, VICTORIA, Illustrating Surface Shape in Volume Data via Principal Direction- Driven 3D Line Integral Convolution, *Computer Graphics Proceedings*, Annual Conference Series (SIGGRAPH 97), pp. 109-116.
- [6] LEVOY, MARC, Display of Surfaces from Volume Data, *IEEE Computer Graphics and Applications*, Vol. 8, No. 3, May 1988, pp. 29-37.
- [7] LORENSEN, W.E. and CLINE, H.E., Marching cubes: A high resolution 3-d surface construction algorithm, *Computer Graphics Proceedings*, Annual Conference Series (SIGGRAPH 1987), pp. 163-169.
- [8] NOORUDDIN, F.S. and GREG TURK, Simplification and Repair of Polygonal Models Using Volumetric Morphology, *Technical Report GIT-GVU-99-37*, Georgia Institute of Technology, 1999.
- [9] RHEINGANS, PENNY, Opacity-modulated Triangular Textures for Irregular Surfaces, *Proceedings of IEEE Visualization '96*, San Francisco, California, Oct. 27 - Nov 1, 1996, pp. 219-225.
- [10] JAREK ROSSIGNAC, ABE MEGHAD, and BENGT-OLAF SCHNEIDER, Interactive Inspection of Solids: Cross-sections and Interferences, *Computer Graphics*, Vol. 26, No. 2 (SIGGRAPH 92), July 1992, pp. 353-360.
- [11] SHADE, JOHNATHAN, S. GORTLER, L. HE and R. SZELISKI, Layered Depth Images, *Computer Graphics Proceedings*, Annual Conference Series (SIGGRAPH 98), pp. 231-242.
- [12] TIEDE, U., K. H. HOHNE, M. BOMANS, A. POMMERT, M. RIEMER, and G. WIEBECKE, Investigation of Medical 3D-Rendering Algorithms, *IEEE Computer Graphics and Applications*, Vol. 10, No. 2, 1990, pp. 41-53.

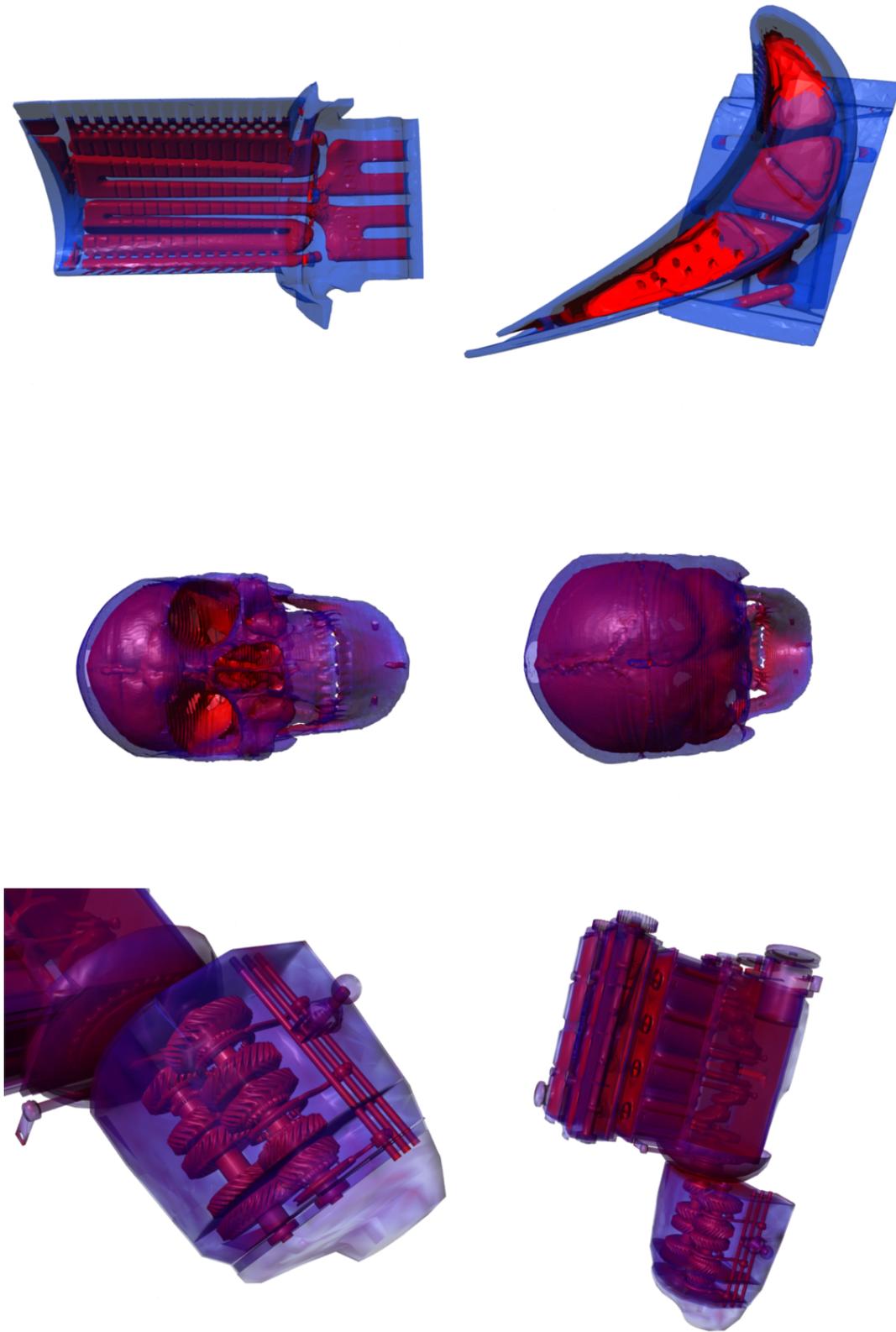
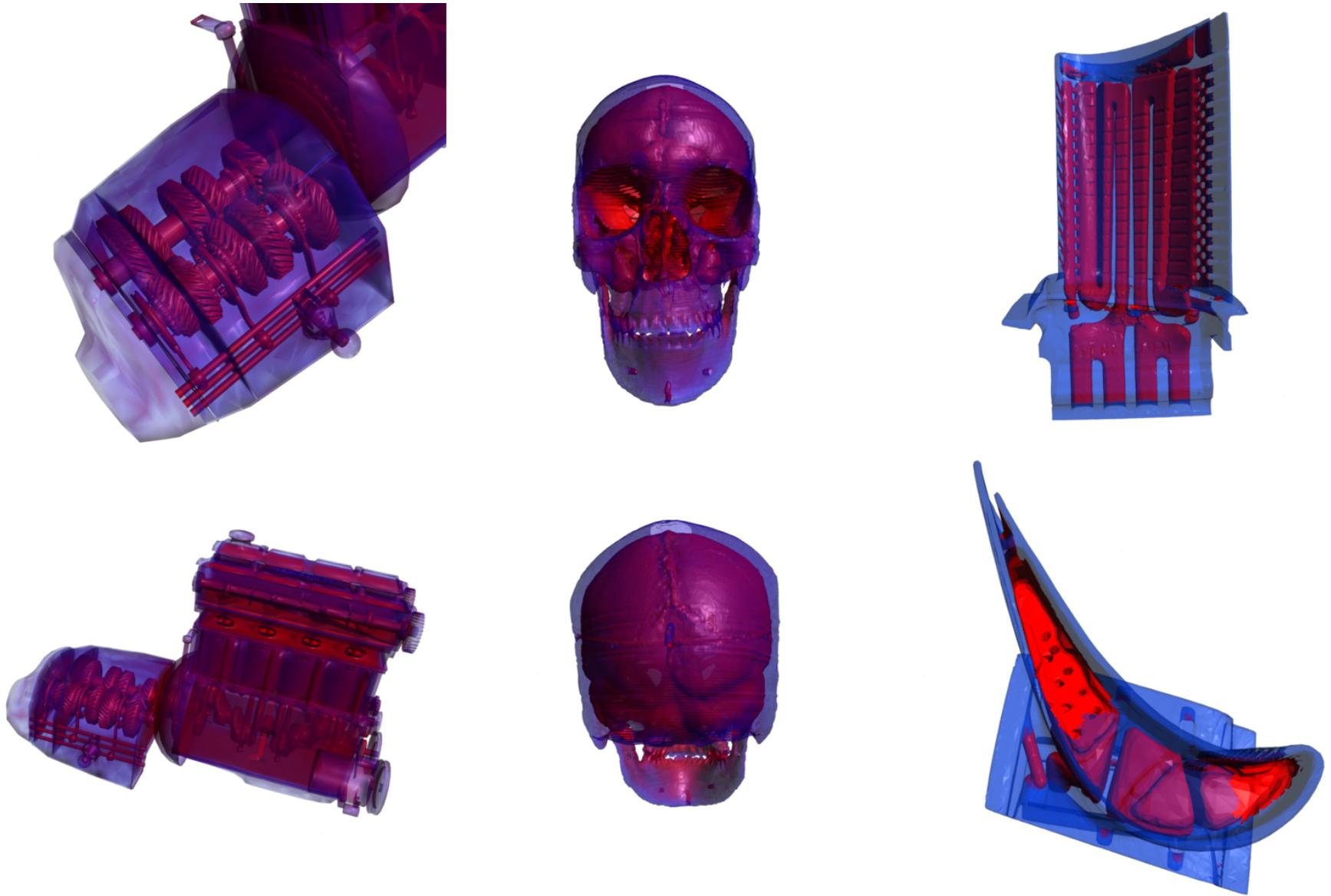


Figure 6: Results of applying our algorithm to the Skull, Motor and Turbine Blade models. A binary classification was performed on the polygons of the blade model. The polygons of the Skull and Motor models had their opacities modulated based on the number of directions from which they were visible.



Color Plate: Renderings of the results of applying our algorithm to the Skull, Motor and Turbine Blade models.