# Volume Sampled Voxelization of Geometric Primitives

Sidney W. Wang and Arie E. Kaufman

Department of Computer Science
State University of New York at Stony Brook
Stony Brook, NY 11794-4400

## Abstract

*We present a 3D antialiasing algorithm for voxel-based geometric models. The technique band-limits the continuous object before sampling it at the desired 3D raster resolution. By precomputing tables of filter values for different types and sizes of geometric objects, the algorithm is very efficient and has a complexity that is linear with the number of voxels generated. The algorithm not only creates voxel models which are free from object space aliasing, but it also incorporates the image space antialiasing information as part of the view independent voxel model. The resulting alias-free voxel models have been used to model synthetic scenes, for discrete ray tracing applications. The discrete ray-traced image is superior in quality to the image generated with a conventional surface-based ray tracer, since silhouettes of objects, shadows, and reflections appear smooth (jaggy-less). In addition, the alias-free models are also suitable for intermixing with sampled datasets, since they can be treated uniformly as one common data representation.*

**Keywords**: Voxelization, Volume Sampling, Discrete Ray Tracing, Filtering

## 1. Introduction

Recent progress in volume rendering technology and voxel-based graphics is apparent, not only for visualizing and analyzing sampled and computed datasets, but also for modeling synthetic scenes as in volume graphics [14]. Modeling a geometric scene in voxel space calls for algorithms that generate from a geometric representation of the scene the set of voxels that "best" approximates the continuous model. These algorithms, called *voxelization* (or 3D scan conversion) algorithms, are applied to a variety of objects with 1D, 2D, and 3D degrees of freedom, that is, curves, surfaces, and solids, respectively. Voxelizing a continuous model into a 3D raster requires a sampling process which determines the value to assign to each element, or voxel, of the 3D raster. Perhaps the most straightforward method of sampling in space is by *point sampling*. Due to its simplicity, the point sampling approach is employed by all of the voxelization algorithms which have appeared in the literature to date [3, 9-11]. In point sampling, we select the voxel center as the representative for each voxel, evaluate the original continuous object at this point, and assign the value of 0 or 1 to the voxel. Because of this binary classification of the voxels, the resolution of the 3D raster ultimately determines the precision of the discrete model. Imprecise modeling results in jagged surfaces known as *object space aliasing* [1, 2]. Aliasing in volume graphics presents greater difficulties than those of 2D raster graphics. Unlike antialiasing of 2D scan-converted graphics, where the main focus is on generating aesthetically pleasing displays, the emphasis in antialiased 3D voxelization, which is the subject of this paper, is on producing alias-free models for various volume graphics applications.

*Volume graphics* [14] is concerned with the synthesis, manipulation, and rendering of volumetric objects, primarily geometric models, stored in a 3D raster of voxels. Volume graphics offers several advantages that are due to the decoupling, uniformity, and atomicity features of the approach. In volume graphics the rendering phase is viewpoint independent and insensitive to scene complexity and object complexity as demonstrated in [19]. It supports effectively Boolean and block operations and constructive solid modeling [13]. When 3D sampled or simulated data is available, such as that generated by medical scanners or scientific simulations, volume graphics is suitable for their representation.

However, the aliasing of the point sampled models causes many of the maladies of voxel-based graphics. For example, in discrete ray tracing [19] and flight simulation [18] applications, the discrete geometric primitives are used to model the synthetic scene. During shading and secondary ray calculations, an essential requirement is the ability to accurately compute the surface normal vector. However, the lack of geometric definition of the surface in discrete voxel representation necessitates the use of discrete shading techniques which estimate the normal from a context of neighboring voxels [20]. The accuracy of discrete normal estimation depends greatly on the smoothness/jaggedness of the discrete surface. Needing accurate normals for spawning secondary rays, Yagel et al. [19] have assumed that ray-object intersection always occurs at voxel center by pre-storing during voxelization the exact surface normal within each voxel. A more obvious effect of aliasing can be seen when detecting discrete ray-object intersection. A ray that barely misses an object in the continuous space might produce an intersection in the discrete space. Similarly, a ray-object intersection in the continuous space might not be detected in the discrete space. These two scenarios are illustrated in Figure 1.

Voxelized geometric primitives are also being utilized in conjunction with sampled data as in medical imaging. For example, geometrically defined objects and sampled CT datasets need to be intermixed and visualized together, such as when a scalpel is superimposed on a CT image, or radiation beams are superimposed on a scanned tumor [12, 15]. A preferred solution is to convert (voxelize) the geometric object into the sampled data representation before intermixing them. The resulting composite dataset is then rendered using one of a variety of volume rendering methods. However, if the geometric primitive is voxelized using point sampling, artifacts may occur in the generated image. The reason lies in the incompatibility of merging a data of binary density (the voxelized primitive) with a data of grey scale density (the sampled CT data).

Another application employing voxelized primitives is Constructive Solid Geometry (CSG). CSG operations such as subtraction, union, and intersection between two voxelized objects are accomplished at the voxel level [16], thereby reducing the original problem of evaluating a CSG tree of such operations down to a Boolean operation between pairs of voxels. However, since the precision of the discrete model is determined by the 3D raster resolution, errors caused by imprecise modeling can accumulate and lead to artifacts.

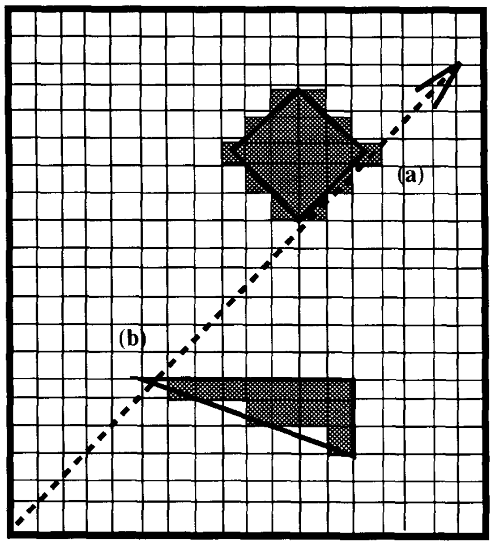


**Figure 1: A ray passing through a volume consisting of two objects. (a) The ray has a false intersection. (b) The ray has a miss intersection.**

The above examples present the complications inherited in the point sampled discrete models in a variety of volume graphics applications. Like pixels in 2D, voxels in 3D may in principle be made as small as desired to increase the accuracy of the discrete representation, thus reducing the aliasing. However, the improvement comes at the price of significantly increasing the memory space and scan conversion time. In this paper we devise an analytical 3D antialiasing technique for modeling continuous geometric primitives in a discrete 3D raster. The resulting discrete model is alias-free and produces high quality 2D image (superior to the image generated with conventional ray tracers) when ray traced with a discrete ray tracer. Furthermore, the antialiased discrete model not only eliminates aliasing in the object space, but it also smoothes out the *image space aliasing* caused by point sampling in screen space, without the need for sub-pixel supersampling or other postprocessing.

## 2. Weighted Volume Sampled Primitives

The main problem with point sampling lies in the fact that only a finite set of points are sampled, and therefore, important features presented on the boundary between the material (the object) and the empty space may be missed. This is why aliased 2D models contain jagged edges and aliased 3D models have jagged surfaces. A commonly used antialiasing method in 2D is simply to point sample the object at a high resolution before filtering (averaging) it down to the desired image resolution [4]. However, this approach merely "hides" the aliasing by making the jagged silhouette less noticeable to the human eyes. While this method is usually adopted in 2D raster graphics, the same technique will not work in 3D, since in volume graphics the main concern is to generate "precise" models for the application, not just for viewing. Another 2D antialiasing approach follows a filtering paradigm [17] by low pass filtering the original signal (the continuous object) before sampling. After prefiltering, point sampling is feasible. In practice, the two stages, prefiltering followed by sampling, are combined into one process which is called *area sampling* in 2D raster graphics [6]. Gupta and Sproull [7] have incorporated the idea of area sampling into efficient 2D scan conversion algorithms of lines and polygons by exploiting precomputed lookup tables.

Our algorithm for generating alias-free 3D models is essentially a generalization of Gupta and Sproull's area sampling algorithm. Instead of performing 2D area prefiltering and sampling, the algorithm prefilters and samples in 3D space, and hence is termed *volume sampling*. Since we know in advance exactly where the samples are taken (at voxel centers), we need only to evaluate the filtered value at each sample point. Filtering is essentially an averaging process; the filtered value located at a given sample $(x,y,z)$ is determined by positioning the filter support centered at that sample and calculating the density contribution within the filter support:

$$\text{density of sample } (x,y,z) =$$

$$\int_{\text{volume of filter support}} h(r) \cdot f(r, \theta, \phi)\, dV \qquad (1)$$

where

$$f(r, \theta, \phi) = \begin{cases} 1 & \text{if } point(r, \theta, \phi) \in primitive \\ 0 & otherwise \end{cases}$$

$$h(r) = filter\ weight\ function$$

and

$$dV = r^2 \cdot \sin\phi\, dr\, d\phi\, d\theta$$

The filter support we selected is a spherical volume with its radius set at three voxel units. The consequences of making the filter support radius larger or smaller will be discussed later. The filter weight function, $h(r)$, is a weight function that specifies the magnitude of importance of each point within the filter support. Since the filter support is radially symmetric with respect to the center of filter, $h(r)$ can be defined as a function of distance from the filter's center along the radial axis. For the ease of implementation and its effectiveness, a Bartlett function, which has its maximum value at the center of the filter and decreases linearly to zero at a distance equal to the filter support radius, is used as our filter weight function. The use of a filter weight function gives the name *weighted volume sampling* to our sampling technique. Throughout the rest of this paper, the terms volume sampling and weighted volume sampling are used interchangeably.

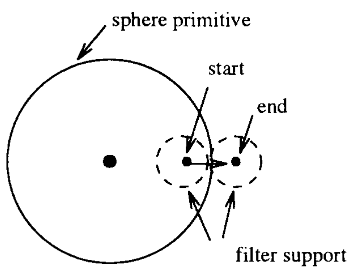
In Equation 1 we have shown how the filtered value of each sample at voxel center is calculated. However, approximating the continuous integral using numerical methods are extremely time consuming. In our implementation, lookup tables are used to assign the filtered value of each voxel rather than compute it on the fly. For each voxel visited in the voxelization algorithm, the distance to the primitive is used as an index into a lookup table of densities. These density values in the table were precomputed using Equation 1. The pseudo code for generating a sphere primitive's lookup table is given in Figure 2.

Since a primitive is usually formed by different types of entities, a lookup table is associated with each type of entity. For example, a polygon primitive consists of faces, edges, and vertices. Hence, three lookup tables are required for the voxelization of a polygon, as illustrated in Figure 3. Then, for a given voxel, one must first identify the entity type to which the voxel belongs; the corresponding lookup table is then used for density assignment.

Additional processing time can be significantly reduced by examining only voxels which are within or located near the continuous primitive instead of a brute force method of examining every voxel in the 3D raster. This can be accomplished easily by utilizing the existing point sampled voxelization algorithms which only visit voxels that are "relevant." Since filtered primitives are thicker than their continuous counterparts, we need to point sample a slightly larger primitive (depending on the filter support radius) to ensure that all essential voxels are visited.

## 3. Benefits of Volume Sampling

In Section 1 we discussed the problems associated with 3D point sampled primitives. In this section, we



```
sphere_tbl ( sphere_rad, filter_rad )
{
    start_pos = sphere_center.x + sphere_rad - filter_rad;
    end_pos  = sphere_center.x + sphere_rad + filter_rad;

    for (filter_pos = start; filter_pos < end; filter_pos += delta_x)
        table [filter_pos] = density of sample (filter_pos);
}
```

**Figure 2: Generating density lookup table for a sphere.**

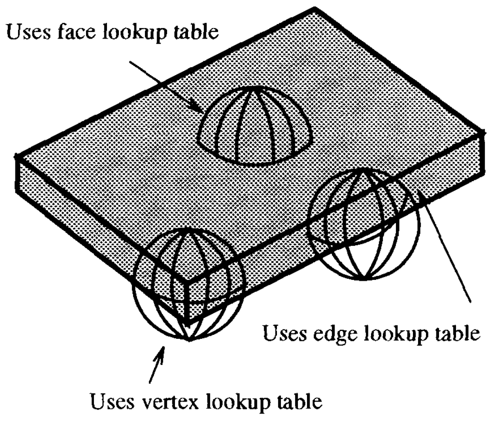


**Figure 3: Filtering a polygon requires three density lookup tables.**

```
curr_pos = ray_origin;
threshold = surface_filtered_density;

while ( curr_pos.density < threshold && In_Volume(curr_pos) )
{
    prev_pos = curr_pos;
    prev_pos.density = curr_pos.density;
    curr_pos += ray_stepping_incr;
    curr_pos.density = Trilinear_Interpolate (curr_pos);
}

if ( In_Volume( curr_pos ) )
    intersection = Binary_Search( prev_pos, curr_pos, threshold);
else
    intersection = NULL;
```

**Figure 4: Ray-object intersection detection algorithm.**

describe how these difficulties are eliminated when alias-free primitives are used instead.

Recall that jagged surfaces of point sampled primitives result in complication during ray-object intersection detection (e.g., Figure 1). With volume sampled primitives, one might think it is impossible to detect the original surface of the primitive, since the prefiltering process smoothed out the original surface. However, by treating the filtered values in the 3D raster as samples in a scalar field and knowing the filtered density value of the primitive's surface, one can detect the original surfaces by thresholding during ray tracing. Hence, neither a false intersection nor an undetected intersection may occur.

The ray-object intersection detection algorithm is described in Figure 4. Basically, at each stepping of the ray, a trilinear interpolation is performed to evaluate the scalar value at that point. If that value is greater than the surface filter density (the threshold value), the surface must pass between the current ray position and the previous ray position. A binary search is then performed to detect the exact position of the ray-object intersection. The level of the binary search can be set according to the ray stepping increment size and the desired fidelity. The intersection detection algorithm assumes that there is only one threshold density value associated with a given object surface. This assumption, however, holds true for spheres only. For other primitives we need to normalize the entity density values, one for each type of entity (see Figure 3), in order for the algorithm to work. By taking advantage of the fact that traversing a discrete ray is much faster than traversing a continuous parametric ray, we can step quickly through the empty space by utilizing a discrete ray, and once the ray reaches any non-transparent voxel, a more accurate continuous ray is used for intersection detection.

When an intersection point is detected, the surface normal at that particular point can be accurately estimated from the context of the neighboring samples by grey-level gradient normal estimation [8]. In our implementation, this is estimated by calculating the density central differences among the six neighboring samples taken exactly one voxel unit above, below, in front, behind, to the left of, and to the right of the exact intersection point. The reason to require that the filter radius size be at least three voxel units is to assure that the neighboring samples contain values for normal estimation. Of course, the larger the selected filter radius size, the larger is the neighborhood available for normal evaluation. But a larger filter smoothes out the sharp features of the original object (e.g., the apex of a cone, the edges and vertices of a polygon). Computing the surface normal on the fly eliminates the need to pre-store the normal during the voxelization process, thus saving memory space. More importantly, we no longer assume that the ray-object intersection always occurs at the voxel center; thus normals are more accurately computed and, for example, secondary rays are spawned in the precise direction.

In volume sampling, the fuzzy (filtered) surface has the property that the density of the surface decreases smoothly from the interior of the object to the empty space. This property of diminishing densities of the filtered surface is essentially the image space antialiasing information that is provided for the ray tracer. Normally, when a continuous object with a well defined surface is ray traced, a ray either hits or misses the object. This binary classification of the pixels results in jaggedness around the silhouette of objects, reflections, and shadows. However, with volume sampling, a ray can approximate the closeness of a miss by comparing the ratio of the current density value to the normalized surface density value while stepping along the ray. A ratio of one or greater represents a "solid" hit, while a ratio between zero and one represents a "close" miss. Therefore, when determining the final color of a pixel, the closeness ratio of a ray-object miss is also considered in the final pixel intensity calculation. For example, a closeness ratio of 0.5 contributes only 50 percent of the object intensity to the ray. The same technique is also used to generate soft shadows by allowing partial blockage of light as rays pierce through the fuzzy silhouette of the object. Thus, the effect of image space antialiasing and soft shadow (shown in Figures 5 and 6) can be accomplished without the need for any extra postprocessing.

Volume sampled primitives also solve the problem of compatibility when merging point sampled primitives with sampled/computed datasets, since the volume sampled primitives are sampled with the same frequency as the sampled/computed dataset, as though the voxelized primitive is generated by a sampling device such as a CT scanner. Hence, the intermixed datasets are treated uniformly as one data representation and can be rendered using any volume rendering method. Figure 7 shows a simulated dataset visualized together with volume sampled mirrors.

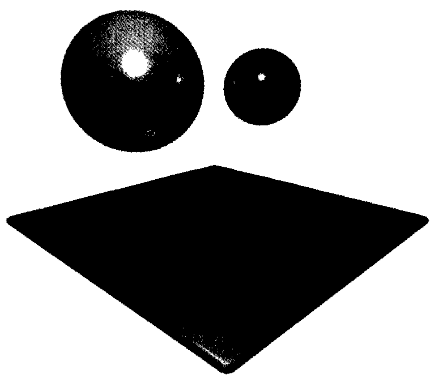


**Figure 6: Volume sampled spheres casted soft shadows on a polygon.**

In addition, volume sampled primitives can function as *matte volumes* [5] for various matting operations such as CSG operations. One example is merging multiple volumes into a single volume using union operation. Another common uses of matte volumes is to perform cut-aways. In Figure 8, a sphere matte volume is used to remove a spherical section of the CT scanned head. Since our matte volumes are fractional instead of binary, the boundaries between the material and the empty space are smooth and continuous.

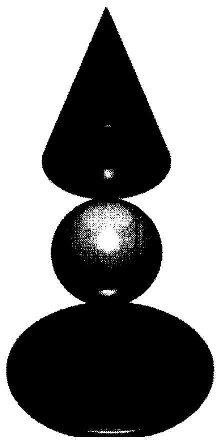


**Figure 5: Volume sampled sphere and disk reflected onto the base of a cone.**

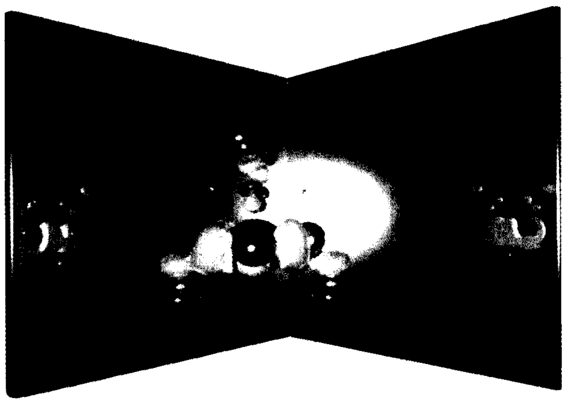


**Figure 7: Simulated molecule reflected onto volume sampled mirrors.**

*(See color plates, p. CP-9.)*

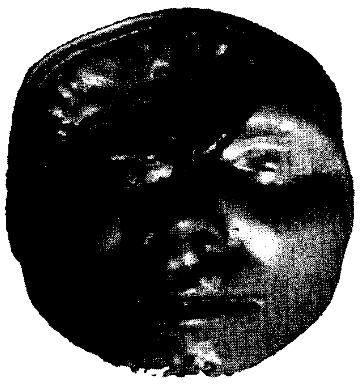


**Figure 8: Portion of the head is removed by subtracting a volume sampled sphere.**

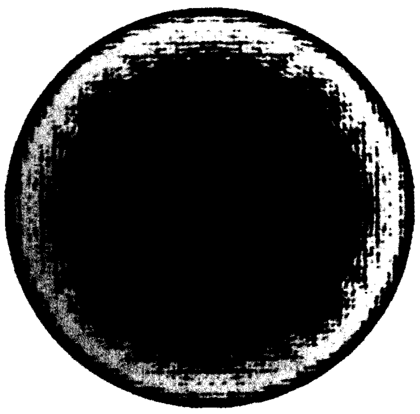


**Figure 9: Error magnitude of ray-object intersection detection. Green represents** $0.0 \le |e| < 0.05$ **voxel units, yellow represents** $0.05 \le |e| < 0.1$**, and red represents** $0.1 \le |e| < 0.2$.

## 4. Accuracy

In this section, we examine the accuracy of our ray-object intersection detection and surface normal estimation. Figures 9 and 10 are color mapped images which indicate the magnitude of errors when compared to the true geometric ray-object intersection and the true geometric surface normal, respectively.

As seen in Figure 9, the accuracy of the ray-object intersection detection degrades for rays that are around the silhouette of the primitive. The reason is because the primitive's density value changes very slowly in the direction that is perpendicular to the surface normal; hence, for a ray located near the silhouette of the primitive, it might intersect the threshold surface value at a small interval rather than a single point. Nevertheless, the maximum error is kept under 0.2 voxel unit.

Recall that our surface normal estimation is achieved through a simple grey-level gradient method by taking the central differences. Figure 10 shows that the maximum normal estimation error is under 1.5 degrees. The result can be improved by employing a more sophisticated normal estimation technique or examining a larger neighborhood. The pattern generated in the normal error image is caused by the quantization error of representing each voxel as an unsigned char rather than a float.

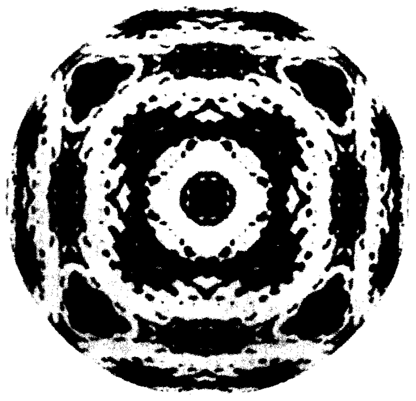


**Figure 10: Error magnitude of surface normal estimation. Green represents** $0.0 \le |e| < 0.5$ **degrees, yellow represents** $0.5 \le |e| < 1.0$**, and red represents** $1.0 \le |e| < 1.5$.

## 5. Conclusion and Future Work

Voxelization of primitives using a weighted volume sampling technique has been presented. The advantages of volume sampled models as compared to point sampled models have been discussed and demonstrated. The ability to incorporate both object space antialiasing and image space antialiasing directly into the voxel model during a preprocessing stage solidifies the potential of volume graphics. We are currently working on a new surface detection algorithm which will not require the surface density values to be normalized; as a result, for a given volume sampled primitive, we will be able to treat a given volume sampled primitive as a "hard" object for modeling synthetic scene in ray tracing application, or as a "soft" object for intermixing with sampled data.

*(See color plates, p. CP-9.)*

## 6. Acknowledgements

This work has been supported by the National Science Foundation under grant CCR-9205047. The images were generated using a modified version of Lisa Sobierajski's ray tracer.

## 7. References

1. Blinn, J. F., "Return of the Jaggy", *IEEE Computer Graphics & Applications*, **9**, 2 (March 1989), 82-89.
2. Blinn, J. F., "What We Need Around Here is More Aliasing", *IEEE Computer Graphics & Applications*, **9**, 1 (January 1989), 75-79.
3. Cohen, D. and Kaufman, A., "Scan-Conversion Algorithms for Linear and Quadratic Objects", in *Volume Visualization*, A. Kaufman, (ed.), IEEE Computer Society Press, Los Alamitos, CA, 1990, 280-301.
4. Crow, F. C., "A Comparison of Antialiasing Techniques", *IEEE Computer Graphics & Applications*, **1**, 1 (January 1981), 40-48.
5. Drebin, R. A., Carpenter, L. and Hanraham, P., "Volume Rendering", *Computer Graphics*, **22**, 4 (August 1988), 65-74.
6. Foley, J., Dam, A., Feiner, S. and Hughes, J., *Computer Graphics: Principles and Practice, Second Edition*, Addison-Wesley, 1990.
7. Gupta, S. and Sproull, R. F., "Filtering Edges for Gray-Scale Displays", *Computer Graphics*, **15**, 3 (August 1981), 1-5.
8. Hoehne, K. H. and Bernstein, R., "Shading 3D Images from CT Using Gray-Level Gradient", *IEEE Transactions on Medical Imaging*, **MI-5**, 1 (March 1986), 45-47.
9. Kaufman, A. and Shimony, E., "3D Scan-Conversion Algorithms for Voxel-Based Graphics", *Proceedings of the 1986 Workshop on Interactive 3D Graphics*, Chapel Hill, NC, October 1986, 45-75.
10. Kaufman, A., "An Algorithm for 3D Scan-Conversion of Polygons", *Proceedings of EUROGRAPHICS'87 Conference*, Amsterdam, The Netherlands, August 1987, 197-208.
11. Kaufman, A., "Efficient Algorithms for 3D Scan-Conversion of Parametric Curves, Surfaces, and Volumes", *Computer Graphics*, **21**, 4 (July 1987), 171-180.
12. Kaufman, A., Yagel, R. and Cohen, D., "Intermixing Surface and Volume Rendering", in *3D Imaging in Medicine. Algorithms, Systems, Applications*, K. H. Hoehne, H. Fuchs and S. M. Pizer, (eds.), Springer-Verlag, 1990, 217-227.
13. Kaufman, A., "The *voxblt* Engine: A Voxel Frame Buffer Processor", in *Advances in Graphics Hardware III*, A. A. M. Kuijk, (ed.), Springer-Verlag, Berlin, 1992, 85-102.
14. Kaufman, A., Cohen, D. and Yagel, R., "Volume Graphics", *Computer*, July 1993.
15. Levoy, M., "A Hybrid Ray Tracer for Rendering Polygon and Volume Data", *IEEE Computer Graphics & Applications*, March 1990, 33-40.
16. Mehl, M. E. and Meagher, D. J., "Geometric Modeling using Octree Encoding", *Computer Garphics and Image Processing*, **19**, 2 (June 1982), 129-147.
17. Wolberg, G., *Digital Image Warping*, IEEE Computer Society Press, 1990.
18. Wright, J. and Hsieh, J., "A Voxel-Based, Forward Projection Algorithm for Rendering Surface and Volumetric Data", *Proceeding Visualization '92*, Boston, MA, October 1992, 340-348.
19. Yagel, R., Cohen, D. and Kaufman, A., "Discrete Ray Tracing", *IEEE Computer Graphics & Applications*, September 1992, 19-28.
20. Yagel, R., Cohen, D. and Kaufman, A., "Normal Estimation in 3D Discrete Space", *The Visual Computer*, June 1992, 278-291.

*Volume Sampled Voxelization of Geometric Primitives*, S.W. Wang and A.E. Kaufman, pp. 78-84.

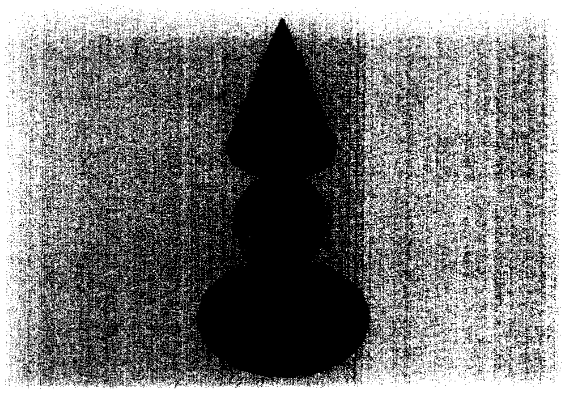


Figure 5: Volume sampled sphere and disk reflected onto the base of a cone.

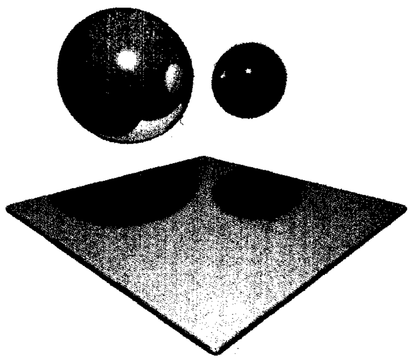


Figure 6: Volume sampled spheres casted soft shadows on a polygon.

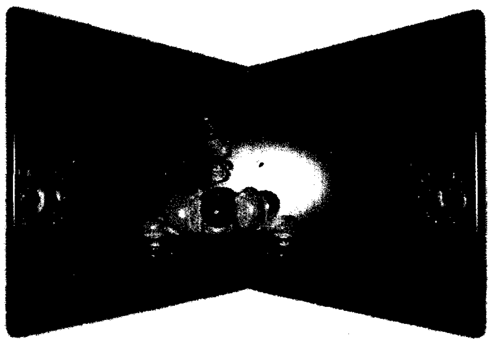


Figure 7: Simulated molecule reflected onto volume sampled mirrors.

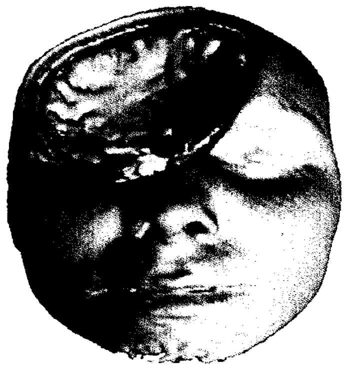


Figure 8: Portion of the head is removed by subtracting a volume sampled sphere.

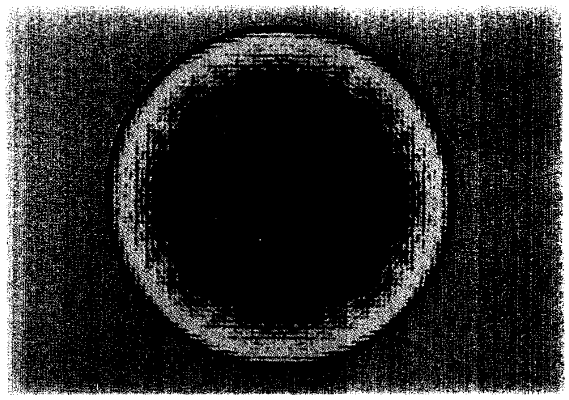


Figure 9: Error magnitude of ray-object intersection detection. Green represents $0.0 \leq |e| < 0.05$ voxel units, yellow represents $0.05 \leq |e| < 0.1$, and red represents $0.1 \leq |e| < 0.2$.

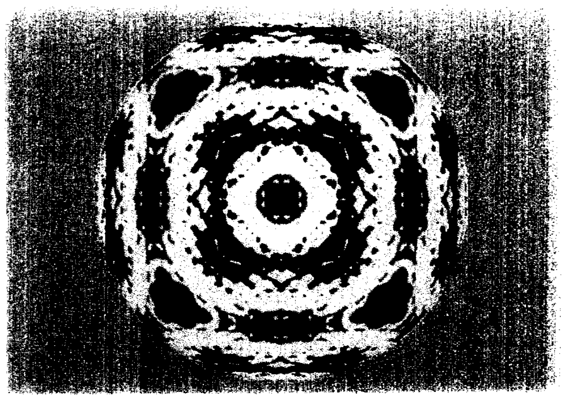


Figure 10: Error magnitude of surface normal estimation. Green represents $0.0 \leq |e| < 0.05$ degrees, yellow represents $0.5 \leq |e| < 1.0$, and red represents $1.0 \leq |e| < 1.5$.