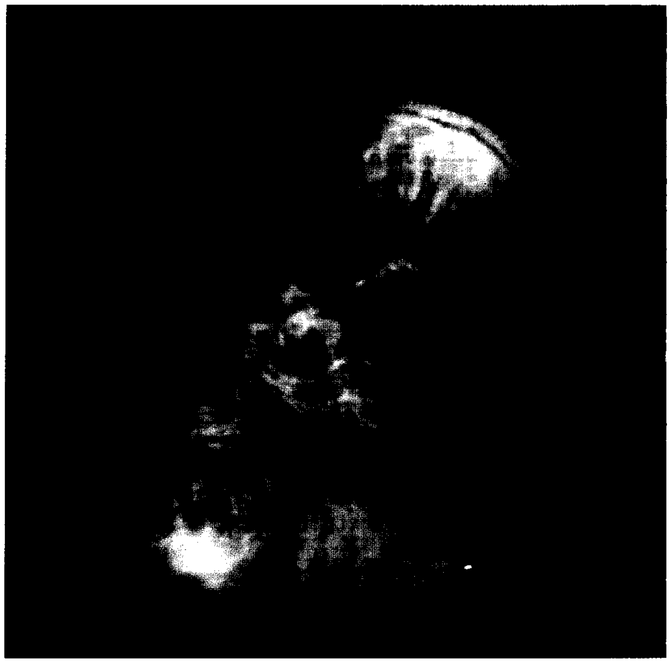
Volume-Sampled 3D Modeling

Sidney W. Wang and Arie E. Kaufman
State University of New York at Stony Brook



This technique creates voxel models that are free from object-space aliasing. Thus it incorporates the image-space antialiasing information as part of the view-independent voxel model.

Volume graphics is an emerging area of computer graphics concerned with the synthesis, manipulation, and rendering of volumetric objects stored in a volume raster of voxels.[1] Typically, this technology samples geometric objects, stores them with their view-independent attributes in the volume raster, and repeatedly projects from the volume raster onto a 2D image raster. This approach is an alternative to conventional surface-based graphics, and its advantages over the latter include view-independent representation, sensitivity to the volume raster resolution but not to object complexity, and suitability for representing sampled and simulated datasets and their mixtures with geometric objects. Further, it supports the visualization of internal structures and lends itself to block operations, hierarchical representations, and constructive solid modeling at the voxel level.

On the other hand, due to the discrete nature of the model representation, volume graphics suffers some disadvantages. It requires substantial amounts of storage space. Moreover, if not properly synthesized, voxelized volume models are imprecise. Thus, they generate jagged surfaces, known as *object-space aliasing*, and significant errors, especially after a sequence of transformations and manipulations is performed on them. (The literature includes detailed accounts of the relative advantages and disadvantages of volume and surface graphics.[1])

Modeling a synthetic scene in volume graphics calls for algorithms that use geometric objects to generate a set of voxels that approximate the continuous object. These algorithms, called *voxelization* (or 3D scan-conversion) algorithms, are applied to a variety of objects such as curves, surfaces, and solids. The goal of our research is to improve the precision of voxelized volume models. In this article, we present a 3D object-space antialiasing technique for volume graphics. Our approach performs antialiasing once—on a 3D view-independent representation—as part of the modeling stage. Unlike antialiasing of 2D scan-converted graphics, where the main focus is on generating aesthetically pleasing displays, antialiasing of 3D voxelized graphics emphasizes the production of alias-free 3D models for various volume graphics manipulations, including but not limited to the generation of aesthetically pleasing displays.

Point-sampled models

Voxelizing a continuous object into a volume raster requires a sampling process that assigns a value to each element, or voxel, of the volume raster. Perhaps the most straightforward method of sampling in space is *point sampling*. Due to its simplicity, all the voxelization algorithms reported in the literature employ this method (for example, see Cohen and Kaufman,[2] Kaufman,[3] and Mokrzycki[4]). Point sampling evaluates the continuous object at the voxel center and assigns the value of 1 when the voxel belongs to the object or 0 otherwise. This binary voxelization scheme means that the resolution of the 3D raster ultimately determines the precision of the discrete model.

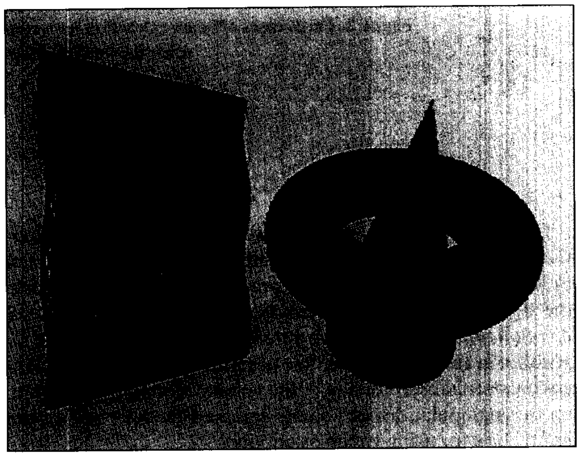


Figure 1. Volumetric ray-tracing image of point-sampled cone and torus reflected on a point-sampled wavy mirror.

The imprecision of point-sampled models causes object-space aliasing that, in turn, causes many of the maladies of voxel-based graphics. For example, discrete ray tracing[5] and voxel-based flight simulation[6] applications use discrete geometric models to model their synthetic scenes. Accurate scene rendering requires accurate computation of surface normal vectors to calculate shading and spawn secondary rays. However, the lack of geometric surface definitions in discrete voxel representation necessitates the use of discrete shading techniques that estimate the normal from a context of neighboring voxels. The accuracy of the discrete normal estimation depends greatly on the smoothness/jaggedness of the discrete surface.

Another obvious effect of aliasing occurs when detecting 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. Figure 1 shows a volumetric ray-traced image of a point-sampled torus and a point-sampled cone reflected on a point-sampled wavy mirror. The image was generated using the VolVis volumetric ray tracer,[7] which employs continuous parametric rays in the discrete 3D volume raster. This type of ray tracing is referred to as *volumetric ray tracing*, in contrast with discrete ray tracing,[5] which uses discrete rays instead.

Some applications use voxelized geometric models in conjunction with sampled data. For example, a medical imaging application might superimpose a scalpel on a 3D reconstructed computed tomography (CT) image or radiation beams on a 3D magnetic resonance imaging (MRI) scan of a tumor. One way to do this is to convert (voxelize) the geometric object into the volumetric sampled data representation before intermixing them. You could then render the resulting composite dataset using one of a variety of volume rendering methods. However, if the geometric object is voxelized using point sampling, artifacts can occur in the generated image. The reason lies in the incompatibility of merging data of binary density (the voxelized model) with data of gray-scale density (the sampled data).

Voxelized primitives lend themselves to various block-based modeling, such as *voxblt* (voxel block transfer) operations. These operations—the 3D counterparts of bitblt operations—support the transfer and manipulation of cuboidal voxel blocks (that is, 3D windows, or "rooms") with a variety of voxel-by-voxel operations between source and destination blocks. This is especially advantageous when the modeling paradigm is constructive solid geometry. CSG operations such as subtraction, union, and intersection between two voxelized objects occur at the voxel level. This reduces the original problem of evaluating a CSG tree of such operations during rendering to a Boolean operation between pairs of voxels during modeling. However, since the 3D raster resolution determines the precision of the discrete model, errors caused by imprecise point-sampled modeling can accumulate and lead to gross artifacts.

The spatial presortedness of the volume raster lends itself to multiresolution hierarchical model representations. These representations approximate the original model by decreasing 3D raster resolution as the hierarchy level increases. They accomplish this by aggregating neighboring voxels (for example, $2 \times 2 \times 2$ neighborhoods) into supervoxels in a pyramid-like hierarchy. To render unbounded scenes and to approach real-time rendering, distant objects are modeled with low 3D raster resolution while objects closer to the viewpoint are modeled with higher resolution (compare with Wright and Hsieh[6]). However, if the model represented at the highest resolution is aliased, the object-space aliasing subsequently propagates through all levels of the resolution hierarchy.

The examples in this section present the complications inherent in point-sampled discrete modeling for a variety of volume graphics manipulations. Like pixels in 2D, voxels in 3D can 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 voxelization time. In the next section, we present an algorithm for generating alias-free geometric primitives using volume sampling. Then we describe how our alias-free models resolve each of the complications associated with point-sampled models.

Volume-sampled models

The main problem with point sampling lies in the finite set of points sampled, which can miss important features present on the boundary between the material (the object) and empty space. This is why aliased 2D models contain jagged edges and aliased 3D models have jagged surfaces. One antialiasing method in 2D is simply to point-sample the object at a high resolution, then average it down to the desired image resolution. However, this approach merely hides the aliasing by making the jagged silhouette less noticeable to the human eye. While this method is frequently adopted in 2D raster graphics, it will not work in 3D, since volume graphics must generate alias-free models for various manipulations, not just for viewing.

Another 2D antialiasing approach uses a low-pass filter on the original signal (the continuous object) before sampling it. In practice, the two stages—prefiltering followed by sampling—are combined into one process, called *area sampling* in 2D raster graphics. Gupta and Sproull[8] incorporated this idea into efficient 2D scan-conversion algorithms for lines and polygons by exploiting precomputed look-up tables.

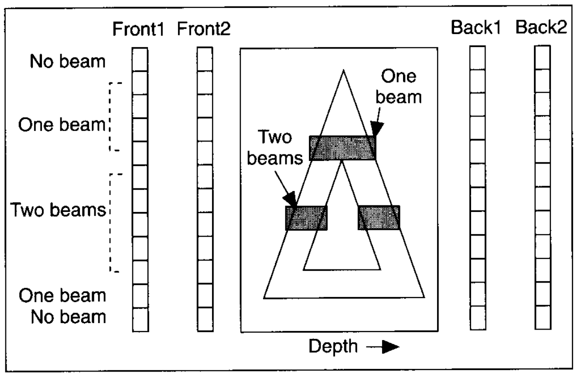


Figure 2. Three cases of beam extraction for a cone using the depth buffers.

Our algorithm for generating alias-free 3D models essentially generalizes Gupta and Sproull's area-sampling algorithm to 3D. Instead of performing 2D area prefiltering and sampling, the algorithm prefilters and samples in 3D space, hence we call it *volume-sampling*. We have detailed the volume-sampling algorithm in an earlier paper.[9] We describe it here briefly, but focus on an enhanced version of the volume-sampling algorithm, on the manipulations of volume-sampled models, and on the motivation for using volume sampling.

For its effectiveness and ease of implementation, we selected a 4D hypercone filter for volume sampling. The hypercone filter has a spherical filter support, and it is weighted such that its maximum contribution occurs at the center of the sphere and attenuates linearly to zero at a distance equal to the radius of the sphere. The filter support should be large enough to filter out all the high frequencies that exceed the Nyquist frequency as determined by the sampling resolution of the volume raster, but also small enough to retain as many of the object's detailed features as possible.

To speed up the process of volume sampling, we use precomputed feature- and object-dependent look-up tables of filtered densities to assign the filtered-density value to each voxel.[9] Thus, the expensive process of performing 3D volume filtering at each voxel is reduced to a look-up table reference. However, it is impossible to precompute feature- and object-dependent tables of densities for all shapes and sizes of primitives. Our implementation includes look-up tables only for a predefined set of geometric primitives of certain sizes. For a primitive that falls between two predefined sizes, we linearly interpolate a value between the two corresponding look-up tables.

We achieve another speedup by using existing point-sampled voxelization algorithms to access those voxels affected by the filtered object directly, instead of traversing the entire volume raster. Unlike point-sampled models, where the outer voxelized surface is at most half a voxel unit away from the outer continuous surface, the outer surface of volume-sampled models is at most the filter radius away from its corresponding continuous outer surface. Hence, to ensure that all essential voxels are visited, we expand the continuous object by the size of the filter radius. Then a point-sampled voxelization algorithm labels the expanded outer surface. Finally, a volume-fill algorithm extracts all interior voxels bounded by the outer voxelized surface. However, in many applications, such as synthetic scene modeling, voxelizing the interior of the continuous model is unnecessary because the application typically renders only the surface. Thus, additional processing time can be saved by treating the continuous object as hollow rather than solid and filtering just its surface.

We developed another technique to extract only those voxels affected by the filtered surface. The idea is to point-sample the outer as well as inner voxelized surfaces bounding the filtered "soft" region, then to use volume-filling to extract all voxels bounded between these surfaces. In general, however, extracting voxels between two voxelized surfaces of an arbitrarily shaped object is nontrivial. Therefore, the algorithm presented here is restricted to convex geometric primitives without cavities, such as cones, cylinders, and spheres. The proposed algorithm can still generate other geometric primitives, such as tori, but the process requires extra care.

The algorithm employs a data structure that consists of a pair of front and back 2D depth buffers, called *front1* and *back1*, respectively, for the outer voxelized surface, and a pair of front and back depth buffers, called *front2* and *back2*, respectively, for the inner voxelized surface. Typically, the depth buffers are arbitrarily placed parallel to one of the principal planes of the volume raster. However, some nonconvex objects might benefit from selecting one specific principal plane. For example, when voxelizing a torus, the depth buffers are placed parallel to the principal plane that is most parallel to the base plane of the torus.

For the sake of discussion, let's assume that the depth buffers are placed parallel to the *xy*-plane. During point-sampled voxelization of the outer surface, each pixel of the front1 and back1 depth buffers is routinely updated to reflect the smallest and largest depth value, respectively. Similarly, the front2 and back2 depth buffers are updated during voxelization of the inner surface. Once the inner and outer surfaces are voxelized, a fast orthogonal beam-filling algorithm extracts all voxels bounded by the two voxelized surfaces. Figure 2 illustrates the three cases of beam extraction for a cone in an analogous 2D diagram.

Table 1 shows volume-sampling times measured on a Silicon Graphics Indigo 2 for a selected set of geometric primitives. The table breaks the processing time into three parts. The first part represents the time needed to update the depth buffers employing the point-sampled voxelization algorithms. The second part indicates the time needed to extract the orthogonal beams from the depth buffers, while the third part is the time spent classifying the extracted voxels to the correct feature-dependent look-up tables and assigning the density values to these voxels. The selected filter support had a radius of two voxel units.

Clearly, as the primitive becomes more complex, the classification process determining which feature a voxel represents becomes more expensive. For example, in generating a volume-sampled box primitive, the majority of processing time is spent on feature classification. This is because each extracted voxel requires point-plane distance calculations to find the distance to the nearest face, edge, or vertex of the box primitive. The calculated distance then becomes an index to the corresponding look-up table.

Table 1. Volume-sampling times (in seconds) for a sphere, a cone, and a box.

Primitive	Interior	Voxel Extraction: Depth Buffers	Voxel Extraction: Filling	Feature Classification for Table Lookup	Total Time
Sphere (radius: 32)	Solid	0.07	0.38	1.67	2.12
	Hollow	0.12	0.14	0.65	0.91
Cone (height: 64; base radius: 32)	Solid	0.69	0.25	2.84	3.78
	Hollow	1.09	0.16	1.70	2.95
Box (dimension: 32×64×32)	Solid	0.07	0.23	4.74	5.05
	Hollow	0.12	0.16	2.76	3.07

Rendering volume-sampled models

Once the filtered scene is represented as a volume raster of filtered values, we can basically treat it as a sampled or simulated volume dataset, such as a 3D medical imaging dataset. Thus, we can employ one of many volume-rendering techniques for image generation—either an image-order approach by casting a ray from each image pixel into object space (for example, see Levoy[10]) or an object-order approach by directly mapping each sample point in object space into image space (see Drebin, Carpenter, and Hanrahan[11]). However, in this article, we discuss only image-order volume-rendering algorithms, specifically surface ray casting[10] and volumetric ray tracing.[7] Note that you can view our volume-sampled model as either a density cloud or a solid object with surface features. This flexible viewing is not achievable through traditional surface rendering.

Both the ray-casting and the volumetric ray-tracing algorithms determine pixel values by casting rays through the image plane to the dataset. However, ray casting casts only primary rays, while ray tracing recursively spawns secondary rays at ray-object intersection points and thus supports the simulation of global illumination phenomena. Detecting ray-surface intersection in a filtered scene is not a trivial task, since the filtered scene no longer includes the geometric surface definitions.

A surface ray caster uses a threshold isosurface value to determine the presence of a surface. This binary classification is a source of image-space aliasing. The reason lies in the high frequencies introduced by the sharp transitions present on binary-classified surfaces. Our method employs a nonbinary surface classification instead, which maps filtered values to opacity factors. Consequently, the smoothness of the band-limited scene in object space carries over into image space, thus eliminating the sharp surface transitions.

Detecting a ray-object intersection is simple with information provided by the surface classifier. Basically, at each sample point along the ray, we perform a trilinear interpolation among surrounding voxels to evaluate the filtered value at that point. Then the classifier assigns the corresponding opacity factor. A pixel color is obtained by compositing the shaded samples along the pixel ray in a front-to-back order, terminating when the accumulated opacity reaches opaque or when the ray exits the dataset.

The method for casting secondary rays and shadow rays for the volumetric ray tracing is the same as for the primary rays. Thus, the silhouettes of objects—as well as their reflections, refractions, and shadows—are jaggy-free. For example, contrast Figure 3 with Figure 1. Needing accurate normals to spawn secondary rays, the ray tracer employs a gray-level gradient normal,[12] estimated from the context of the neighboring samples, for spawning reflective and transmitted rays. In our implementation using VolVis,[7] we use central differences to estimate the components of the normal vector. By not performing any geometric ray-object intersection or geometric surface normal calculation, we save the bulk of the rendering time.

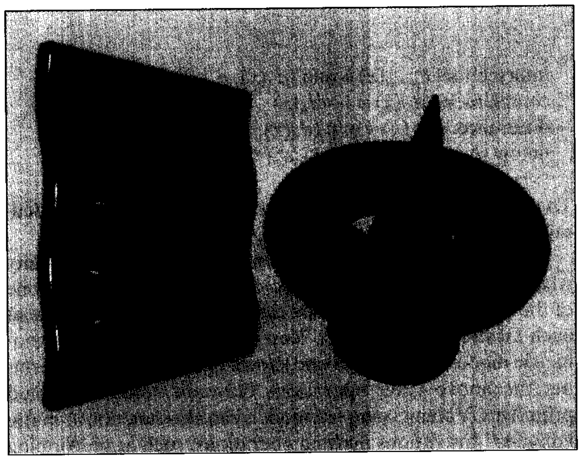


Figure 3. Volumetric ray-tracing image of volume-sampled torus and cone reflected on a volume-sampled wavy mirror.

Manipulating volume-sampled models

Recall that manipulating point-sampled models presents several problems. For example, during CSG model construction, the errors caused by imprecise modeling can accumulate. Another problem arises from the data-type incompatibility of visualizing a point-sampled model with a scanned or simulated dataset. Volume-sampled models eliminate these difficulties.

Our model is a density function $d(x)$ over R^3, where $d = 1$ inside the filtered object, $d = 0$ outside the filtered object, and $0 < d < 1$ within the soft region of the filtered surface. Hence, the Boolean operations of CSG or voxblt applied to volume-sampled models are analogous to those of fuzzy set theory (for example, see Dubois and Prade[13]). Some common operations between two objects A and B are defined as follows:

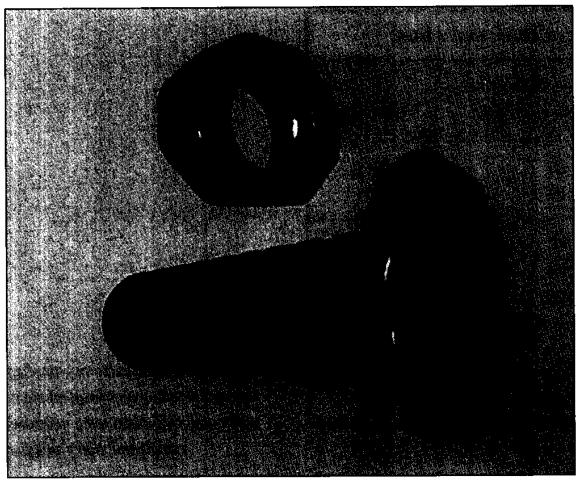


Figure 4. Bolt and nut generated by a sequence of CSG operations on hexagonal, cylindrical, and helix primitives.

intersection: $d_{A \cap B}(x) \equiv \min(d_A(x), d_B(x))$
complement: $d_{\bar{A}}(x) \equiv 1 - d_A(x)$
difference: $d_{A-B}(x) \equiv \min(d_A(x), 1 - d_B(x))$
union: $d_{A \cup B}(x) \equiv \max(d_A(x), d_B(x))$

The only law of set theory that is no longer true is the excluded-middle law (that is, $A \cap \bar{A} \neq \phi$ and $A \cup \bar{A} \neq$ universe).

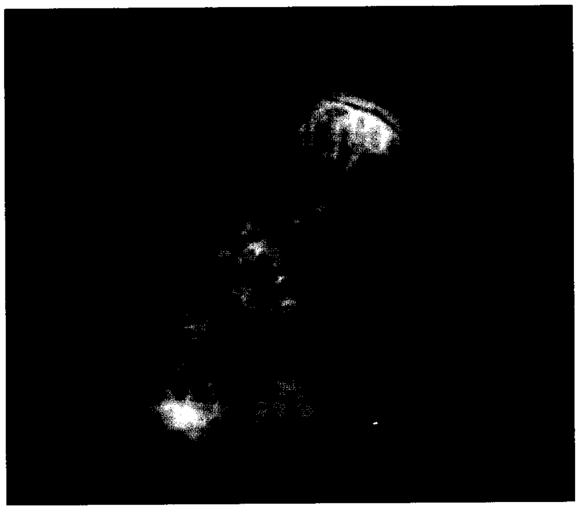
The use of the min and max functions causes discontinuity where the soft regions of the two objects meet, since only one of the two overlapping objects can define the density value at each location in the region. You can generate complex geometric models, such as the one shown in Figure 4, by performing the above CSG operations between volume-sampled primitives. Volume-sampled models can also function as matte volumes[11] for various matting operations, such as performing cutaways and using the union operation to merge multiple volumes into a single volume. However, to preserve continuity on the cutaway boundaries between the material and the empty space, you should use an alternative set of Boolean operators based on algebraic sum and algebraic product:[13,14]

intersection: $d_{A \cap B}(x) \equiv d_A(x)\, d_B(x)$
complement: $d_{\bar{A}}(x) \equiv 1 - d_A(x)$
difference: $d_{A-B}(x) \equiv d_A(x) - d_A(x)\, d_B(x)$
union: $d_{A \cup B}(x) \equiv d_A(x) + d_B(x) - d_A(x)\, d_B(x)$

Unlike the min and max operators, algebraic sum and product operators result in $A \cup A \neq A$. Consequently, if the modeling uses sweeping, for example, the resulting model is sensitive to the sampling rate of the swept path. For instance, sweeping a volume-sampled sphere around a circle with two revolutions produces a different volume-sampled torus from sweeping with just one revolution.

For intermixing manipulations, we use the second set of Boolean operators to determine the density values of the overlapping region between a volume-sampled primitive and a sampled/computed dataset. For example, to visualize a radiation beam superimposed on a human MRI head, we can first model the radiation beam using a volume-sampled primitive (for example, a cylinder); then we can position the synthesized radiation beam at the desired location within the MRI dataset. During volume rendering, as a ray enters the overlapping region, the union of the density values from the two datasets determines the density values along that portion of the ray. Hence, unifying the intermixed models in a common data representation eliminates the need for a special-purpose renderer. Figure 5 shows the image resulting from an intermixing. Our VolVis system[7] treats each voxel as a density value in a scalar field, and it can handle multiple, even overlapping, volumes. Thus, we can freely position and rotate the synthesized radiation beam, regardless of the MRI dataset's position and orientation.

Figure 5. Intermixing of a volume-sampled cylinder with an MRI head using the union operation.



Finally, volume-sampled models also resolve the issue of propagating object-space aliasing during multiresolution hierarchy construction. The direct correspondence between the size of the filter support and the resolution of the volume raster leads to an ideal alias-free hierarchical model representation. The base of the hierarchy contains the most detailed, highest resolution of the object, and the top contains the blurriest, low-resolution version.

Volume-sampled modeling does require extra care to ensure that adequate filtering is applied for the given 3D raster resolution at each level of the hierarchy. We do this by widening the filter support, thus decreasing the raster resolution as you move up the hierarchy. Object-space aliasing is, consequently, eliminated for all levels of the resolution hierarchy. Figure 6 shows four different hierarchy levels rendered. The appropriate hierarchy level is selected according to distance from the viewpoint. Objects close to the viewpoint are modeled with a high level of detail, while distant objects are modeled with a low-resolution representation. Figure 7 illustrates the mapping of projection depth to hierarchy level. For a resolution that lies between two levels of the hierarchy, the data value is determined at each of the two adjacent levels. Then it is bilinearly interpolated between them to avoid spatial and especially temporal discontinuities during animation.

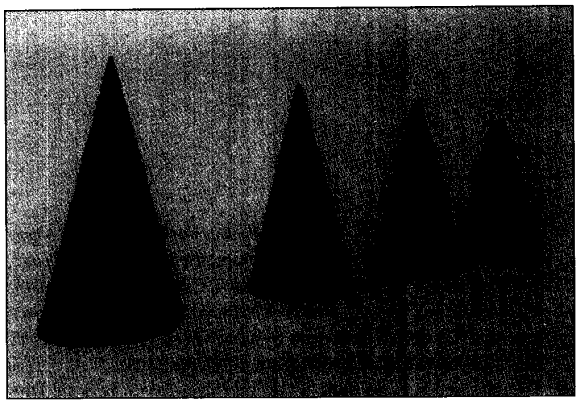
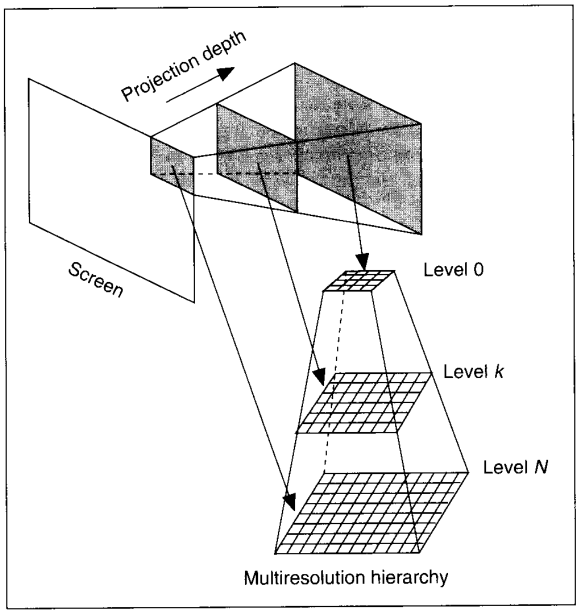


Figure 6. Rendering of four different resolution representations of the same object.

Figure 7. Projection depths are mapped to their corresponding hierarchy levels.



Results

We conducted tests to determine the accuracy of our ray-object intersection detection and surface normal estimation as well as the rendering performance. Figures 8a and 8b are color-mapped images that indicate the magnitude of errors when compared with the true geometric ray-object intersection and the true geometric surface normal, respectively. In both cases, errors are linearly mapped to the colors red and yellow, where red indicates the largest error magnitude and yellow indicates no error.

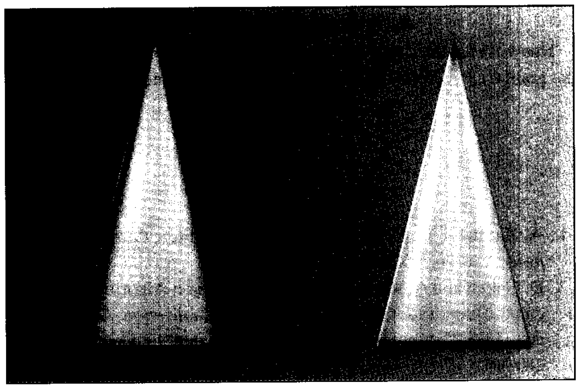
As Figure 8a shows, the accuracy of ray-object intersection detection degrades for rays around the silhouette of the geometric object. This is because the accumulation of opacity values along the silhouette of the volume-sampled object during volume rendering can cause the ray to terminate early. Thus, false intersection occurs. The maximum error in surface detection is under 0.2 voxel unit for a volume-sampled cone.

Recall that we estimate our surface norm through a gray-level gradient method by taking the central differences. Figure 8b shows that the magnitude of error is greatest at locations near the apex and circumference of the cone base. This is because volume filtering smoothed out the object's sharp features. Nevertheless, normal estimation error is under 1.5 degrees for areas other than the apex and the circumference. We can improve these results by employing a more sophisticated normal estimation technique or examining a larger neighborhood.

Although the 3D volume-filtering process is expensive, it is performed just once as a preprocessing stage; the resulting alias-free models are stored as part of the database. The view-independent database can then be rendered repeatedly using volume rendering or volumetric ray tracing. Furthermore, by carrying the smoothness of the volume-sampled 3D model over into image space, this approach achieves image space anti-aliasing without increasing the number of pixel rays or performing image-space filtering.

Now refer back to the examples of volumetric ray-traced and volume-rendered images of volume-sampled models in Figures 3 and 4, respectively. We performed all renderings on a Silicon Graphics Indigo 2 using a modified version of the VolVis volumetric ray tracer. The scene illustrated in Figure 3 consists of three volumes, with 3D raster resolutions of 30 × 150 × 125, 165 × 45 × 165, and 70 × 140 × 70. We used one generation of reflection rays and one light source for ray tracing. For image resolutions of 300 × 300 and 600 × 600, the ray tracing times are 35 seconds and 141 seconds, respectively. The volume-rendered image of Figure 4 consists of two volumes with 3D raster resolutions of 70 × 40 × 60 and 70 × 125 × 65. Ray-casting time for a 300 × 300 resolution image took 21 seconds and for a 600 × 600 image took 87 seconds.

Figure 8. Error magnitudes mapped linearly from yellow to red, where yellow represents no error and red represents maximum error: (a) error in surface detection as compared to the true geometric ray-object intersection and (b) error in surface normal estimation as compared to the true geometric surface normal.



We made no attempt to optimize the ray-casting or ray-tracing codes. For example, we could improve the rendering performances by precomputing and storing in the volume rasters such view-independent shading attributes as diffuse reflections and shadow determinations. Furthermore, the flexibility of handling multiple volumes in our VolVis rendering system adds overhead to our rendering time by treating these volumes first as bounding boxes for objects in the scene before utilizing the volumetric ray tracer.

Future directions

We are investigating an alternative volume-sampling approach that will eliminate the need for volume-filling and feature-classification processes. The idea is to voxelize the continuous object once, using point sampling. Then, for each voxel visited, the decision variables of the voxelization algorithm function as indices to look-up tables that store the density values of the neighboring voxels. This is similar to Gupta and Sproull's approach in 2D line scan conversion. In essence, the densities of the voxels visited during point-sampled voxelization are splatted in 3D space and distributed to their neighboring voxels.

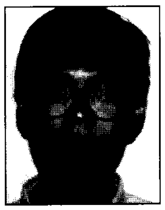
Furthermore, current work focuses on the rendering aspect of volume-sampled models. By carrying the smoothness of the 3D volume-sampled model from object space into its 2D projection in image space, our technique creates not only voxel models that are free from object-space aliasing but also images that are free from image-space aliasing. We are also investigating the benefits of volume-sampled models for approximating the effect of penumbra shadows and fuzzy reflections. ❑

Acknowledgments

This work has been supported by the National Science Foundation under grant CCR-9205047 and by the Department of Energy PICS grant.

References

1. A. Kaufman, D. Cohen, and R. Yagel, "Volume Graphics," *Computer*, Vol. 26, No. 7, July 1993, pp. 51-64.
2. D. Cohen and A. Kaufman, "Scan-Conversion Algorithms for Linear and Quadratic Objects," in *Volume Visualization*, A. Kaufman, ed., IEEE Computer Society Press, Los Alamitos, Calif., 1990, pp. 280-301.
3. A. Kaufman, "Efficient Algorithms for 3D Scan-Conversion of Parametric Curves, Surfaces, and Volumes," *Computer Graphics* (Proc. Siggraph), Vol. 21, No. 4, 1987, pp. 171-180.
4. W. Mokrzycki, "Algorithms of Discretization of Algebraic Spatial Curves on Homogeneous Cubical Grids," *Computer & Graphics*, Vol. 12, Nos. 3/4, 1988, pp. 477-487.
5. R. Yagel, D. Cohen, and A. Kaufman, "Discrete Ray Tracing," *IEEE CG&A*, Vol. 12, No. 5, Sept. 1992, pp. 19-28.
6. J. Wright and J. Hsieh, "A Voxel-Based, Forward-Project Algorithm for Rendering Surface and Volumetric Data," *Proc. Visualization 92*, IEEE CS Press, Los Alamitos, Calif., 1992, pp. 340-348.
7. R. Avila, L. Sobierajski, and A. Kaufman, "Towards a Comprehensive Volume Visualization System," *Proc. Visualization 92*, IEEE CS Press, Los Alamitos, Calif., 1992, pp. 13-20.
8. S. Gupta and R.F. Sproull, "Filtering Edges for Gray-Scale Displays," *Computer Graphics* (Proc. Siggraph), Vol. 15, No. 3, Aug. 1981, pp. 1-5.
9. S.W. Wang and A. Kaufman, "Volume-Sampled Voxelization of Geometric Primitives," *Proc. Visualization 93*, IEEE CS Press, Los Alamitos, Calif., 1993, pp. 78-84.
10. M. Levoy, "Display of Surfaces from Volume Data," *IEEE CG&A*, Vol. 8, No. 5, May 1988, pp. 29-37.
11. R.A. Drebin, L. Carpenter, and P. Hanrahan, "Volume Rendering," *Computer Graphics* (Proc. Siggraph), Vol. 22, No. 4, Aug. 1988, pp. 65-74.
12. K.H. Hoehne and R. Bernstein, "Shading 3D Images from CT Using Gray-Level Gradient," *IEEE Trans. Medical Imaging*, Vol. MI-4, No. 1, Mar. 1986, pp. 45-47.
13. D. Dubois and H. Prade, *Fuzzy Sets and Systems: Theory and Applications*, Academic Press, New York, 1980.
14. K. Perlin and E.M. Hoffert, "Hypertexture," *Computer Graphics*, Vol. 23, No. 3, July 1989, pp. 253-262.



Sidney W. Wang is a PhD candidate in computer science at State University of New York at Stony Brook. His research interests include volume visualization, voxel-based modeling, and volume manipulation. Wang received his BS and MS degrees in electrical engineering and computer science from Johns Hopkins University in 1987 and 1988, respectively.

Arie E. Kaufman is a guest editor of this issue, and his biography appears with the Guest Editor's Introduction.

Readers can contact Kaufman at the Department of Computer Science, State University of New York at Stony Brook, Stony Brook, NY 11794-4400, e-mail ari@cs.sunysb.edu.