3D Discrete Imaging

Volume Synthesis

Arie E. Kaufman
Computer Science Department
State University of New York at Stony Brook
Stony Brook, NY 11794-4400, USA
ari@cs.sunysb.edu http://www.cs.sunysb.edu/~ari

Abstract: This paper is a survey of volume synthesis techniques which are part of the field of volume graphics. It focuses specifically on the use of voxel representation and volumetric techniques for traditional geometric applications of computer graphics.

1. Introduction

Volume data are 3D entities that may have information inside them, might not consist of surfaces, or might be too voluminous to be represented geometrically. *Volume visualization* is a method of extracting meaningful information from volumetric data using interactive graphics and imaging. It is concerned with volume data representation, modeling, manipulation, and rendering [16]. Volume data are obtained by sampling (e.g., CT, MRI, ultrasound, confocal microscope), simulation (e.g., running on a supercomputer), or modeling. Recently, volume modeling (volume synthesis) has been flourishing, where many traditional geometric graphics applications, such as CAD and flight simulation, have been exploiting the advantages of volume techniques for modeling, manipulation, and visualization. This approach is called *volume graphics.*

Over the years many techniques have been developed to visualize 3D data. Since methods for displaying geometric primitives were already well-established, most of the early methods involve approximating a surface contained within the data using geometric primitives. When volumetric data are visualized using surface rendering, a dimension of information is essentially lost. In response to this, volume rendering techniques were developed that attempt to capture the entire 3D data in a single 2D image. Volume rendering conveys more information than surface rendering, but at the cost of increased algorithm complexity, and consequently increased rendering times. To improve interactivity in volume rendering, many optimization methods as well as several special-purpose volume rendering machines have been developed.

Sec. 2 introduces volume data and volume rendering. Volume graphics is presented in Sec. 3, followed by voxelization (Sec. 4), fundamentals of discrete topology (Sec. 5), point and volume sampling (Secs. 6, 7), texture mapping (Sec. 8), amorphous phenomena (Sec. 9), block operations and constructive solid modeling (Sec. 10), and volume sculpting (Sec. 11). Then, volume graphics is contrasted with surface graphics (Sec. 12), and the features and weaknesses of volume graphics are discussed (Secs. 14, 15).

2. Volume Rendering

Volumetric data is typically a set S of samples (x, y, z, v), representing the value v of some property of the data, at a 3D location (x, y, z). If the value is simply a 0 or a 1, with a value of 0 indicating background and a value of 1 indicating the object, then the data is referred to as binary data. The data may instead be multivalued, with the value representing some measurable property of the data, including, e.g., color, density, or heat. The value v may even be a vector, representing, e.g., velocity at each location.

In general, the samples may be taken at purely random locations in space, but in most cases S is isotropic containing samples taken at regularly spaced intervals along three orthogonal axes. When the spacing along each axis is a constant, but there exist three different spacing constants for the three axes, S is anisotropic. Since the set of samples is defined on a regular grid, a 3D array (called also *volume buffer, cubic frame buffer, 3D raster*) is typically used to store the values, with the element location indicating position of the sample on the grid. Thus, S is referred to as the array of values $S(x, y, z)$, which is defined only at grid locations. Alternatively, either rectilinear, curvilinear (structured), or unstructured grids are employed (e.g., [30]). In a *rectilinear* grid the cells are axis-aligned, but grid spacings are arbitrary. When such a grid has been non-linearly transformed while preserving the grid topology, the grid becomes *curvilinear.* Otherwise the grid is called *unstructured* or *irregular.* An unstructured volume data is a collection of cells whose connectivity has to be specified explicitly. These cells can be of arbitrary shape such as tetrahedra, hexahedra, or prisms.

The array S defines the value of some measured property of the data at discrete locations. A function $f(x, y, z)$ is defined over R^3 describing the value at any continuous location. The function $f(x, y, z) = S(x, y, z)$ if (x, y, z) is a grid location; otherwise $f(x, y, z)$ approximates the value at a location (x, y, z) by applying some interpolation function to S. The simplest function is *zero-order interpolation*, which is just a nearest-neighbor function. With this method there is a region of constant value around each sample in S. The region of constant value surrounding each sample is known as a *voxel* with each voxel being a rectangular cuboid having 6 faces, 12 edges, and 8 corners.

Higher-order interpolation functions can also be used to define $f(x, y, z)$ between sample points. One common interpolation function is a piecewise function known as *first-order interpolation*, or *trilinear interpolation.* With this interpolation function, the value is assumed to vary linearly along directions parallel to the major axes. Let the point P lie at location (x_p, y_p, z_p) within the regular hexahedron, known as a *cell,* defined by samples A through H. For simplicity, let the distance between samples in all three directions be 1, with sample A at $(0, 0, 0)$ with a value of v_A, and sample H at $(1, 1, 1)$ with a value of v_H. The value v_P, according to trilinear interpolation, is then:

$$\begin{aligned} v_P = {} & v_A\,(1 - x_p)(1 - y_p)(1 - z_p) + v_E\,(1 - x_p)(1 - y_p)\,z_p\ + \\ & v_B\ x_p\,(1 - y_p)(1 - z_p) + v_F\,x_p\,(1 - y_p)\,z_p\ + \\ & v_C\,(1 - x_p)\,y_p\,(1 - z_p) + v_G\,(1 - x_p)\,y_p\ z_p\ + \\ & v_D\,x_p\,y_p\,(1 - z_p) + v_H\,x_p\,y_p\ z_p \end{aligned} \tag{1}$$

In general, A is at some location (x_A, y_A, z_A), and H is at (x_H, y_H, z_H). In this case, x_p in Eq. 1 would be replaced by $(x_p - x_A)/(x_H - x_A)$, with similar terms for y_p and z_p.

Representing a surface contained within a volumetric dataset using geometric primitives is useful in some applications, however it has several drawbacks. First, geometric primitives can only approximate surfaces within the data. Adequate approximations may require an excessive amount of geometric primitives. Thus, a trade-off must be made between accuracy and space. Second, since only a surface representation is used, much of the information contained within the data is lost. Also, amorphous phenomena, such as clouds, fog, and fire cannot be adequately represented using surfaces.

Volume rendering is the process of creating a 2D image directly from a 3D volume. Volume rendering can be achieved using an *object-order*, an *image-order*, or a *domain-based* technique. Object-order volume rendering uses a *forward mapping* scheme whereby the volume data is mapped onto the image plane [5]. Such a projection can be accomplished by traversing the data samples either in a back-to-front or a front-to-back order, projecting each sample onto the image plane. One such algorithm is the splatting algorithm [37] generating an approximated but smooth rendering. In image-order algorithms, a *backward mapping* scheme is used, whereby rays are cast from each pixel in the image plane through the volume to determine the pixel value. Ray casting [20, 32, 33], discrete ray tracing [39], and volumetric ray tracing [29] are examples of image-based algorithms. In a domain-based technique the spatial volume data is first transformed into an alternative domain, such as compression [8, 25, 40], frequency [7, 22, 31], or wavelet [24, 36], and then a projection is generated from that domain.

Of particular interest is the image-order discrete ray-tracing [39]. Instead of traversing a continuous ray and determining the closest data sample for each step with zero-order interpolation [32], a discrete representation of the ray is traversed. This discrete ray is generated using a 3D line scan-conversion (voxelization) algorithm (see Sec. 6). For each pixel in the image plane, a ray is cast in the direction of the viewing ray. This ray is voxelized and the recursive contribution from each voxel along the path is contributes to the final pixel value, similar to common recursive ray tracing.

3. Volume Graphics

The 3D raster representation seems to be more natural for empirical imagery than for geometric objects, due to its ability to represent interiors and digital samples. Nonetheless, its advantages are also attracting traditional surface-based applications that deal with synthetic scenes of geometric models. The geometric model is *voxelized (3D scan-converted)* into a set of voxels that "best" approximate the model. Each of the voxels is then stored in the volume buffer together with its pre-computed view-independent attributes. The voxelized model can be either binary [2, 12-14] (see Sec. 6) or volume sampled [34] (see Sec. 7), which generates alias-free density voxelization of the model. Some surface-based application examples are the rendering of fractals, hypertextures, fur, gases, CAD models, and terrain for flight simulators [1, 18, 38]. Furthermore, in many applications involving sampled data, such as medical imaging, the data need to be visualized along with synthetic objects, as with scalpels, prosthesis, injection needles, radiation beams, and isodose surfaces. These objects can be voxelized and mixed with the sampled organ in the voxel buffer [15].

Volume graphics [18] is concerned with the synthesis, modeling, manipulation, and rendering of volumetric geometric objects, stored in a volume buffer. Unlike volume visualization which focuses on sampled and computed data, volume graphics is concerned with modeled geometric scenes commonly represented in a volume buffer. As an approach, volume graphics has the potential to greatly advance the field of 3D graphics by offering an alternative to traditional surface graphics.

4. Voxelization

An indispensable stage in volume graphics is the synthesis of voxel-represented objects from their geometric representation. This stage, called *voxelization*, converts geometric objects from their continuous geometric representation into a set of voxels that "best" approximates the continuous object. As this process mimics the scan-conversion process that pixelizes (rasterizes) 2D objects, it is referred to as *3D scan-conversion*. In 2D rasterization the pixels are directly drawn onto the screen to be visualized. However, the voxelization process does not render the voxels but merely generates a database of the discrete digitization of the continuous object.

Intuitively, a proper voxelization would simply "select" all voxels which are met (if only partially) by the object body. Although this approach could be satisfactory in some cases, the objects it generates are commonly too coarse and include excess voxels. For example, when a 2D curve is rasterized into a connected sequence of pixels, the discrete curve does not "cover" the entire continuous curve, but it is connected and concisely and successfully "separates" both "sides" of the curve [3].

One practical meaning of separation is apparent when a voxelized scene is rendered by casting discrete rays from the image plane to the scene. The penetration of the background voxels (simulating the discrete ray traversal) through the voxelized surface causes the appearance of a hole in the final image of the surface. Another type of error might occur when a 3D flooding algorithm is employed either to fill an object or to measure its volume, surface area, or other properties. In this case the nonseparability of the surface causes a leakage of the flood through the discrete surface.

Unfortunately, the extension of the 2D definition of separation to 3D and to voxel surfaces is not straightforward since voxelized surfaces cannot be defined as an ordered sequence of voxels and a voxel on the surface does not have a specific number of adjacent surface voxels. Furthermore, there are important topological issues, such as the separation of both sides of a surface, which cannot be well-defined by 2D terminology. The theory that deals with these topological issues is called *3D discrete topology*. We sketch below a few informal definitions used in this field.

5. Fundamentals of 3D Discrete Topology

The 3D discrete space is a set of integral grid points in 3D Euclidean space defined by their Cartesian coordinates (x, y, z). A voxel is a unit cube centered at the integral grid point. The voxel value is mapped onto $\{0,1\}$: the voxels assigned "1" are the "black" voxels representing opaque objects, and those assigned "0" are the "white" voxels representing transparent background. In Sec. 7 we describe non-binary approaches where the voxel value is mapped onto the interval $[0,1]$ representing either partial coverage, variable densities, or graded opacities. Due to its larger dynamic range of values, this approach supports 3D antialiasing and thus higher quality rendering.

Two voxels are *26-adjacent* if they share either a vertex, an edge, or a face. Every voxel has 26 such adjacent voxels: 8 share a vertex with the center voxel, 12 share an edge, and 6 share a face. Accordingly, face-sharing voxels are defined as *6-adjacent*, and edge-sharing and face-sharing voxels are defined as *18-adjacent*. The prefix N is used to define adjacency, where $N = 6, 18, 26$. A sequence of voxels having the same value (e.g., black) is called an *N-path* if all consecutive pairs are N-adjacent. A set of voxels W is *N-connected* if there is an N-path between every pair of voxels in W. An *N-connected component* is a maximal N-connected set.

Given a 2D discrete 8-connected black curve, there are sequences of 8-connected white pixels (8-components) that pass from one side of the black component to its other side without intersecting it. This phenomenon is a discrete disagreement with the continuous case where a closed curve cannot be penetrated without intersecting it. To avoid such a scenario, it has been the convention to define "opposite" types of connectivity for the white and black sets. "Opposite" types in 2D space are 4 and 8, while in 3D 6 is "opposite" to 26 or to 18. Assume that a voxel space, denoted by Σ, includes one subset of "black" voxels S. If $\Sigma - S$ is not N-connected, that is, $\Sigma - S$ consists of at least two white N-connected components, then S is said to be *N-separating* in Σ. Loosely speaking, in 2D, an 8-connected black path that divides the white pixels into two groups is 4-separating and a 4-connected black path that divides the white pixels into two groups is 8-separating. There are no analogous results in 3D space.

Let W be an N-separating surface. A voxel $p \in W$ is an *N-simple voxel* if $W - p$ is still N-separating. An N-separating surface is called *N-minimal* if it does not contain any N-simple voxel. A *cover* of a continuous surface is a set of voxels such that every point of the continuous surface lies in a voxel of the cover. A cover is a *minimal cover* if none of its subsets is also a cover. The cover property is essential in applications that employ space subdivision for fast ray tracing [10]. The subspaces (voxels) which contain objects have to be identified along the traced ray. Note that a cover is not necessarily separating, while on the other hand it may include simple voxels. In fact, even a minimal cover is not necessarily N-minimal for any N [3].

6. Binary Voxelization

A simple technique for the digitization of solids is spatial enumeration which employs point or cell classification in either an exhaustive fashion or by recursive subdivision [19]. However, this approach is computationally expensive and thus inappropriate for medium or high resolutions. Instead, objects should be directly voxelized, preferably generating an N-separating, N-minimal, and covering set, where N is application dependent. The voxelization algorithms should follow the same paradigm as 2D scan-conversion: should be incremental, accurate, use simple arithmetic (preferably integer only), and have a complexity that is linear with the number of voxels generated.

The literature on voxelization is small. Danielsson [4] and Mokrzycki [23] developed independently similar 3D curve algorithms where the curve is defined by the intersection of two implicit surfaces. Voxelization algorithms have been developed for 3D lines, 3D circles, and a variety of surfaces and solids, including polygons, polyhedra, and quadric objects [12]. Efficient algorithms have been developed for voxelizing polygons using an integer-based decision mechanism embedded within a scan-line algorithm [14], parametric curves, surfaces, and volumes using an integer-based forward differencing technique [13], and quadric objects such as cylinders, spheres, and cones using "weaving" algorithms by which a discrete circle/line sweeps along a discrete circle/line [2]. Fig. 1 includes examples of voxelized polygons, boxes, and cylinders.

In discrete ray casting, a ray is discretized (binary voxelized) into a 6-, 18-, or 26-connected line, and the voxels along this line are considered in determining the final pixel value. If a surface projection is required, the line is traversed until the first voxel of the surface is encountered. This voxel is then shaded and the resulting color is stored in the pixel. 6-connected paths (6-paths) contain almost twice as many voxels as 26-paths, so an image created using 26-paths would require less computation, but a 26-path may miss an intersection that would be detected using a 6-path.

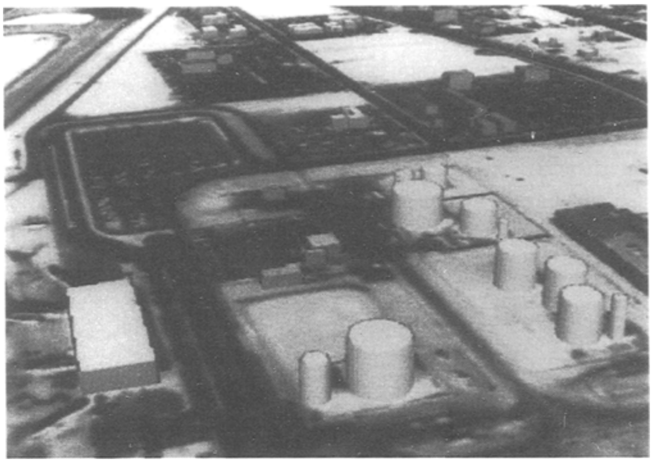

Figure 1: Voxelized terrain enhanced with photo mapping of satellite images. Buildings are synthetic binary voxel models raised on top of the terrain.

7. Volume Sampling

Binary voxelization generates topologically and geometrically consistent models, but exhibit object space aliasing. These algorithms have used a sampling method called *point sampling*. In point sampling, the continuous object is evaluated at the voxel center, and 0 or 1 is assigned to the voxel. Because of this binary classification, 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* (see Fig. 1). In this section, 3D object-space antialiasing is presented. It performs 3D antialiasing once, on a 3D view-independent representation, as part of modeling. Unlike antialiasing of 2D scan-converted graphics, where the focus is on generating aesthetically pleasing displays, the emphasis in antialiased 3D voxelization is on producing alias-free 3D models that are stored in the view-independent volume buffer and may be used also to generate aesthetically pleasing displays.

To reduce object space aliasing, *volume sampling* techniques have been developed [34], which estimate the density contribution of the geometric objects to the voxels. The density of a voxel is attenuated by a filter weight function which is proportional to the distance between the voxel center and the geometric object. Precomputed lookup tables of densities for a predefined geometric primitives can be used to select the density value of voxels visited by the binary voxelization algorithm. Alternatively, each such voxel is "splatted" to its 3D neighbors using a similar filter.

Since the voxelized geometric objects are represented as 3D rasters of densities, they can essentially be treated as sampled or simulated volume data, and volume rendering or volumetric global illumination can be employed. One primary advantage of this approach is that the smoothness of the volume-sampled objects is carried from object space over into its 2D projection in image space. Hence, silhouettes, reflections, and shadows are smooth. Furthermore, by not performing any geometric ray-object intersections or surface normal calculations, the bulk of the rendering time is saved. In addition, constructive solid geometry (CSG) operations between two volume-sampled geometric models are accomplished at the voxel level, thereby reducing the problem of evaluating a CSG tree of such operations down to a fuzzy Boolean operation between pairs of non-binary voxels [34] (see Sec. 10). Volume-sampled models are also suitable for mixing with sampled or simulated datasets, since they can be treated uniformly as one common data representation. Furthermore, volume-sampled models lend themselves to alias-free multi-resolution hierarchy construction [34].

8. Texture Mapping

Objects are commonly enhanced with texture mapping, photo mapping, environment mapping, or solid texturing. Texture mapping is implemented during the last stage of the rendering pipeline, and its complexity is proportional to the object complexity. In volume graphics, however, texture mapping is performed during the voxelization stage, and the texture color is stored in each voxel in the volume buffer.

In photo mapping six orthogonal photographs of the real object are projected onto the voxelized object. Once this mapping is applied, it is stored within the voxels during voxelization, and therefore does not degrade the rendering performance. Texture and photo mapping are viewpoint independent, implying that once the texture is stored as part of the voxel, the mapping need not be repeated. This feature is exploited, e.g., by voxel-based flight simulators (see Fig. 1) and in CAD systems (see Fig. 2).

A central feature of volumetric representation is that, unlike surface representation, it is capable of representing inner structures of objects, which can be revealed with appropriate manipulation and rendering techniques. This capability is essential for the exploration of sampled or simulated objects. Synthetic objects are also likely to be solid rather than hollow. One method for modeling various solid types is solid texturing, in which a function or a 3D map models the color of an object in 3D (see Fig. 2). During the voxelization phase each voxel belonging to the object is assigned a value by the texturing function or the 3D map. This value is then stored as part of the voxel information, and it is recomputed for every change in the rendering parameters.

9. Amorphous Phenomena

While translucent objects can be represented by surface methods, these methods cannot efficiently support the modeling and rendering of amorphous phenomena (e.g., clouds, fire, smoke) that are volumetric in nature and lack any tangible surfaces. A common modeling and rendering approach is based on a function that, for any input point in 3D, calculates some object features such as density, reflectivity, or color. These functions can then be rendered by ray casting. Examples for the use of this or similar techniques are the rendering of fractals, hypertextures, fur, and gases.

The process of function evaluation at every sample point is repeated for every image generated. In contrast, the volumetric approach allows the pre-computation of these functions at every grid point of the volume buffer. The resulting volume can then be rendered from multiple viewpoints without recomputing the modeling function. As in other volume graphics techniques, accuracy is traded for speed, due to the resolution limit. Instead of accurately computing the function at each sample point, some type of interpolation from the precomputed grid values is employed.

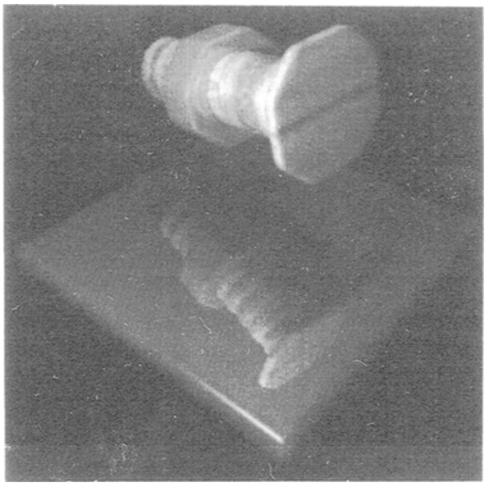

Figure 2: Volume-sampled bolt and nut generated by CSG operations on helix cylindrical, and hexagonal primitives, reflected on a volume-sampled mirror.

10. Block Operations and Constructive Solid Modeling

The presortedness of the volume buffer naturally lends itself to grouping operations. For example, multi-resolution volume hierarchy can support time-critical and space-critical volume graphics applications. The basic idea is similar to that of level-of-detail surface rendering, in which the perceptual importance of a given object in the scene determines its appropriate level-of-detail representation. For example, the terrain image shown in Fig. 1 was generated by the voxel-based Hughes Aircraft Co. flight simulator [38]. It simulates a flight over voxel-represented terrain enhanced with satellite or aerial photo mapping with additional synthetic raised objects, such as buildings, trees, vehicles, aircraft, clouds and the like. Parts of the terrain close to the observer are rendered at high resolution which decreases towards the horizon.

One simple level-of-detail approach is the 3D "mip-map" [21, 28], where every level of the hierarchy is formed by averaging eight voxels from the previous level. A better approach is based on sampling theory, in which an object is modeled with a sequence of alias-free volume buffers at different resolutions using volume-sampled voxelization [11]. To accomplish this, high frequencies are filtered out by applying an ideal low-pass filter (*sinc*) with infinite support. In practice, the ideal filter is approximated by filters with finite support. Low sampling resolution of the volume buffer corresponds to a lower Nyquist frequency, and therefore requires a low-pass filter with wider support for good approximation. As one moves up the hierarchy, low-pass filters with wider and wider support are applied. Compared to the level-of-detail hierarchy in surface graphics, the multi-resolution volume buffers are easy to generate and to spatially correspond neighboring levels, and are free of object space aliasing. Furthermore, arbitrary resolutions can be generated, and errors caused by a non-ideal filter do not propagate and accumulate from level to level. Depending on the required speed and accuracy, a variety of low-pass filters (zero order, cubic, Gaussian) can be applied.

An intrinsic characteristic of the volume buffer is that adjacent objects in the scene are also represented by neighboring memory cells. Therefore, rasters lend themselves to various meaningful grouping-based operations, such as *bitblt* in 2D, or *voxblt* in 3D [17]. These include transfer of volume buffer rectangular blocks (cuboids) while supporting voxel-by-voxel operations between source and destination blocks. Block operations add a variety of modeling capabilities which aid in the task of image synthesis and form the basis for the efficient implementation of a 3D "room manager", which is the extension of window management to the third dimension.

Since the volume buffer lends itself to Boolean operations that can be performed on a voxel-by-voxel basis during the voxelization stage, it is advantageous to use CSG as the modeling paradigm. Subtraction, union, and intersection operations between two voxelized objects are accomplished at the voxel level, thereby reducing the problem of evaluating a CSG tree during rendering time down to a 1D Boolean operation between pairs of voxels during a preprocessing stage.

For two point-sampled binary objects the operations of CSG or *voxblt* are trivially defined. However, the Boolean operations applied to volume-sampled models are analogous to those of fuzzy set theory (cf. [6]). The volume-sampled model is a density function $d(x)$ over R^3, where d is 1 inside and 0 outside the object, and $0 < d < 1$ within the "soft" region of the filtered surface. The common operations, intersection, complement, difference, and union, between two objects A and B are:

$$d_{A \cap B}(x) \equiv \min(\, d_A(x), d_B(x)) \qquad d_{\bar{A}}(x) \equiv 1 - d_A(x) \tag{2}$$

$$d_{A-B}(x) \equiv \min(d_A(x), 1 - d_B(x)) \qquad d_{A \cup B}(x) \equiv \max(\, d_A(x), d_B(x)) \tag{3}$$

However, the excluded-middle law (i.e., $A \cap \bar{A} \neq \phi$ and $A \cup \bar{A} \neq \mathit{Universe}$) is no longer true. The min and max functions cause discontinuity where the soft regions of the two objects meet, since the value at each location in the region is determined solely by one of the two overlapping objects. Complex geometric models can be generated by performing the CSG operations in Eqs. 2-3 between volume-sampled primitives (see Fig. 2). Volume-sampled models can also function as matte volumes [5] for various matting operations, such as performing cut-aways and merging multiple volumes using the union operation. However, in order to preserve continuity on the cut-away boundaries between the material and the empty space, one should use an alternative set of Boolean operators based on algebraic sum and product [6, 26] :

$$d_{A \cap B}(x) \equiv d_A(x)\, d_B(x) \qquad d_{\bar{A}}(x) \equiv 1 - d_A(x) \tag{4}$$

$$d_{A-B}(x) \equiv d_A(x) - d_A(x)\, d_B(x) \qquad d_{A \cup B}(x) \equiv d_A(x) + d_B(x) - d_A(x)\, d_B(x) \tag{5}$$

Unlike the min and max operators, algebraic sum and product operators result in $A \cup A \neq A$, which is undesirable. A consequence, e.g., is that during modeling via sweeping, the resulting model is sensitive to the sampling rate of the sweep [34].

11. Volume Sculpting

Surface-based sculpting has been studied extensively, while volume sculpting has been recently introduced for clay or wax-like sculptures [9] and for detailed sculpting [35]. The latter approach is a free-form interactive modeling technique based on the metaphor of sculpting and painting a voxel-based solid material, such as a block of marble or wood. There are two motivations for this approach. First, modeling topologically complex and highly-detailed objects are still difficult in most CAD systems. Second, sculpting has shown to be useful in visualization applications. For example, scientists and physicians often need to explore the inner structures of their simulated or sampled datasets by gradually removing material.

Real-time interaction could be achieved in this approach. The actions of sculpting (e.g., carving, sawing) and painting are localized in the volume buffer, and thus localized rendering is employed to reproject only those pixels that are affected. Thus, real-time interaction can be achieved. Carving uses a pre-existing volume-sampled tool to chip or chisel the object bit by bit. Since both the object and the tool are represented as independent volumes, the tool is positioned with respect to the object and a Boolean subtraction between the two volumes is performed. Sawing removes a whole chunk of material at once, much like a carpenter sawing off a piece of wood. Unlike carving, sawing requires generating the volume-sampled tool on-the-fly. To prevent object-space aliasing and to achieve interactive speed, 3D splatting is employed.

12. Surface Graphics vs. Volume Graphics

Contemporary 3D graphics has been employing an object-based approach at the expense of maintaining a display list of geometric objects and regenerating the frame-buffer after every change in the scene or the view. This approach, termed *surface graphics*, is supported by powerful geometry engines. Surface graphics strikingly resembles vector graphics that prevailed in the sixties and seventies. Like vector graphics, surface graphics represents the scene as a set of geometric primitives kept in a display list. In surface graphics, these primitives are transformed, mapped to screen coordinates, and then scan-converted into pixels. Any change to the scene, viewing, or shading parameters requires repeating this process. Like vector graphics that did not support painting the interior of 2D objects, surface graphics generates merely the surfaces of 3D objects and does not support the rendering of their interior.

Instead of a display list maintained by surface graphics, volume graphics employs a 3D volume buffer as a medium for representation and manipulation. A 3D scene is discretized earlier in the image generation sequence, and the resulting 3D discrete form is used as a database of the scene for manipulation and rendering purposes, which in effect decouples discretization from rendering. Furthermore, all objects are converted into one uniform meta-object – the voxel. Each voxel is atomic and represents the information about, at most, one object that resides in that voxel. Volume graphics offers similar benefits to surface graphics, with several advantages that are due to the decoupling, uniformity, and atomicity features. The rendering phase is insensitive to scene and object complexities, since all objects have been pre-voxelized with their view-independent attributes into a finite size volume buffer, and rendering performance depends mainly on the volume buffer resolution. Thus, volume graphics is particularly attractive for large scenes and for objects that are hard to render using conventional graphics, such as high-order curved surfaces and fractals, and those with texture and/or antialiasing. It supports Boolean and block operations and CSG. When 3D sampled or simulated data are used, such as that generated by medical scanners (see Fig. 3) or scientific simulations, volume graphic is suitable for their representation or their

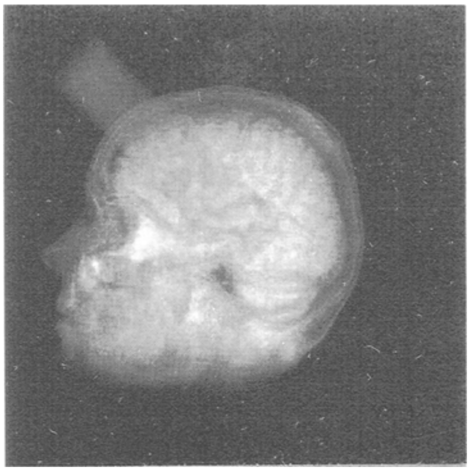

Figure 3: Union operation between volume-sampled cylinder and MRI head.

compatible intemixing with geometric objects (see Fig. 3). It is capable of representing amorphous phenomena and both the interior and exterior of 3D objects.

Several weaknesses of volume graphics are related to the discrete nature of the representation; transformations and shading, for instance, are performed in discrete space. In addition, this approach requires substantial amounts of space (e.g., 256MB for 512^3 2B/voxel) and specialized processing. However, discrete artifacts can be alleviated in ways similar to 2D techniques, computer memories are significantly decreasing in price and size and increasing in speed, and *volume engines*, capable of rendering volumes in true real time of 30 frames/sec, are emerging [16; Ch. 6, 27].

Table 1 contrasts vector graphics with raster graphics. A primary appeal of raster graphics is that it decouples image generation from screen refresh, thus making the refresh insensitive to scene and object complexities. In addition, the raster form lends itself to block operations, such as *bitblt* and quadtree. Raster graphics is also suitable for 2D digital images, and thus provides the ideal environment for mixing digital images with synthetic graphics. Unlike vector graphics, raster graphics provides the capability to present shaded and textured surfaces, as well as line drawings. These advantages, coupled with advances in hardware, have led raster graphics to supersede vector graphics. The main weaknesses of raster graphics are the large memory and processing power it requires, and the discrete nature of the image. These difficulties delayed the full acceptance of raster graphics until the late seventies. In addition, the discrete nature of rasters make them less suitable for geometric operations such as transformations and accurate measurements, and once discretized object notion is lost.

The same appeal that drove the evolution of computer graphics from vector graphics to raster graphics, once the memory and processing power became available, is driving a variety of applications from a surface-based to a volume-based approach. Naturally, this trend first appeared in applications involving sampled or simulated 3D data, such as

Table 1: Comparison between vector graphics and raster graphics and between surface graphics and volume graphics.

2D	Vector Graphics	Raster Graphics
Scene/object complexity	−	+
Block operations	−	+
Sampled data	−	+
Interior	−	+
Memory and processing	+	−
Aliasing	+	−
Transformations	+	−
Objects	+	−
3D	Surface Graphics	Volume Graphics

3D medical imaging and scientific visualization, in which datasets are already in volumetric form. These diverse empirical applications of volume visualization still provide a major driving force for advances in volume graphics.

The comparison in Table 1 between vector and raster graphics strikingly resembles a comparison between surface and volume graphics. Actually, Table 1 itself is used also to contrast surface and volume graphics (see [18]).

13. Conclusions

Several of the concepts and methods of volume visualization have been presented. Although volumetric representations and visualization seem more natural for sampled or simulated datasets, their advantages are also attracting traditional geometric-based applications. This trend implies an expanding role for volume visualization, and it has the potential to revolutionize the field of computer graphics, by providing an alternative to surface graphics, called volume graphics. As summarized in Table 1, volume graphics has advantages over surface graphics by being viewpoint independent, insensitive to scene and object complexities, and lending itself to the realization of block operations, CSG modeling, and hierarchical representation. It is suitable for the representation of sampled or simulated datasets and their intermixing with geometric objects, and it supports the display of internal structures. The problems associated with the volume representation – memory size, processing time, aliasing, and lack of geometric representation – echo problems encountered when raster graphics emerged as an alternative technology to vector graphics, and can be alleviated in similar ways.

The progress so far in volume graphics, in hardware, and memory systems, coupled with the desire to reveal the inner structures of volumetric objects, suggest that volume visualization and volume graphics may develop into major trends in computer graphics. Just as raster graphics in the seventies superseded vector graphics for visualizing surfaces, volume graphics has the potential to supersede surface graphics for handling and visualizing volumes as well as for modeling and rendering synthetic surfaces.

Acknowledgments: Thanks are due to Lisa Sobierajski, Rick Avila, Roni Yagel, Dany Cohen, Sid Wang, Taosong He, Hanspeter Pfister, and Lichan Hong for their help. This work has been supported by NSF grants CCR-9205047 and MIP-9527694. Fig. 1 is courtesy of Hughes Aircraft Co. The scene has been voxelized using voxelization algorithms, a voxel-based modeler, and a photo-mapper developed at SUNY Stony Brook. The MRI head data in Fig. 3 is courtesy of Siemens. For the *VolVis* software send email to: volvis@cs.sunysb.edu.

14. References

1. Cohen, D. and Shaked, A., "Photo-Realistic Imaging of Digital Terrain", *Computer Graphics Forum*, **12**, 3 (September 1993), 363-374.
2. Cohen, D. and Kaufman, A., "Scan Conversion Algorithms for Linear and Quadratic Objects", In [16], pp. 280-301.
3. Cohen-Or, D. and Kaufman, A., "Fundamentals of Surface Voxelization", *CVGIP: Graphics Models and Image Processing*, **56**, 6 (November 1995), 453-461.
4. Danielsson, P. E., "Incremental Curve Generation", *IEEE Transactions on Computers*, **C-19**, (1970), 783-793.
5. Drebin, R. A., Carpenter, L. and Hanrahan, P., "Volume Rendering", *Computer Graphics (Proc. SIGGRAPH)*, **22**, 4 (August 1988), 65-74.
6. Dubois, D. and Prade, H., *Fuzzy Sets and Systems: Theory and Applications*, Academic Press, 1980.
7. Dunne, S., Napel, S. and Rutt, B., "Fast Reprojection of Volume Data", *Proceedings of the 1st Conference on Visualization in Biomedical Computing*, Atlanta, GA, 1990, 11-18.
8. Fowler, J. and Yagel, R., "Lossless Compression of Volume Data", *Proceedings Symposium on Volume Visualization*, Washington, DC, October 1994, 43-50.
9. Galyean, T. A. and Hughes, J. F., "Sculpting: An Interactive Volumetric Medeling Technique", *Computer Graphics*, **25**, 4 (July 1991), 267-274.
10. Glassner, A. S., "Space Subdivision for Fast Ray Tracing", *IEEE Computer Graphics and Applications*, **4**, 10 (October 1984), 15-22.
11. He, T., Hong, L., Kaufman, A., Varshney, A. and Wang, S., "Voxel-Based Object Simplification", *IEEE Visualization '95 Proceedings*, October 1995, 296-303.
12. Kaufman, A. and Shimony, E., "3D Scan-Conversion Algorithms for Voxel-Based Graphics", *Workshop on Interactive 3D Graphics*, Chapel Hill, NC, Oct. 1986, 45-76.
13. Kaufman, A., "Efficient Algorithms for 3D Scan-Conversion of Parametric Curves, Surfaces, and Volumes", *Computer Graphics*, **21**, 4 (July 1987), 171-179.
14. Kaufman, A., "An Algorithm for 3D Scan-Conversion of Polygons", *Proc. EUROGRAPHICS'87*, Amsterdam, Netherlands, August 1987, 197-208.
15. 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.), June 1990, 217-227.
16. Kaufman, A., *Volume Visualization*, IEEE CS Press Tutorial, Los Alamitos, CA, 1991.
17. 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.
18. Kaufman, A., Cohen, D. and Yagel, R., "Volume Graphics", *Computer*, **26**, 7 (1993), 51-64.

19. Lee, Y. and Requicha, A., "Algorithms for Computing the Volume and Other Integral Properties of Solids", *Comm. ACM*, **25**, 9 (Sept. 1982), 635-650.

20. Levoy, M., "Display of Surfaces from Volume Data", *Computer Graphics and Applications*, **8**, 5 (May 1988), 29-37.

21. Levoy, M. and Whitaker, R., "Gaze-Directed Volume Rendering", *Computer Graphics (Proc. 1990 Symposium on Interactive 3D Graphics)*, **24**, 2 (March 1990), 217-223.

22. Malzbender, T., "Fourier Volume Rendering", *Trans. on Graphics*, **12**, 3 (1993), 233-250.

23. Mokrzycki, W., "Algorithms of Discretization of Algebraic Spatial Curves on Homogeneous Cubical Grids", *Computers & Graphics*, **12**, 3/4 (1988), 477-487.

24. Muraki, S., "Volume Data and Wavelet Transform", *IEEE Computer Graphics & Applications*, **13**, 4 (July 1993), 50-56.

25. Ning, P. and Hesselink, L., "Fast Volume Rendering of Compressed Data", *Visualization '93 Proceedings*, October 1993, 11-18.

26. Perlin, K. and Hoffert, E., "Hypertexture", *Computer Graphics*, **23**, 3 (1989), 253-262.

27. Pfister, H. and Kaufman, A., "Cube-4: A Scalable Architecture for Real-Time Volume Rendering", *Volume Visualization Symp. Proc.*, San Francisco, CA, Oct, 1996.

28. Sakas, G. and Hartig, J., "Interactive Visualization of Large Scalar Voxel Fields", *Proceedings Visualization '92*, Boston, MA, October 1992, 29-36.

29. Sobierajski, L. and Kaufman, A., "Volumetric Ray Tracing", *Volume Visualization Symposium Proceedings*, Washington, DC, October 1994, 11-18.

30. Speray, D. and Kennon, S., "Volume Probes: Interactive Data Exploration on Arbitrary Grids", *Computer Graphics*, **24**, 5 (November 1990), 5-12.

31. Totsuka, T. and Levoy, M., "Frequency Domain Volume Rendering", *Computer Graphics (Proc. SIGGRAPH)*, 1993, 271-278.

32. Tuy, H. K. and Tuy, L. T., "Direct 2-D Display of 3-D Objects", *IEEE Computer Graphics & Applications*, **4**, 10 (November 1984), 29-33.

33. Upson, C. and Keeler, M., "V-BUFFER: Visible Volume Rendering", *Computer Graphics (Proc. SIGGRAPH)*, 1988, 59-64.

34. Wang, S. and Kaufman, A., "Volume-Sampled 3D Modeling", *IEEE Computer Graphics & Applications*, **14**, 5 (September 1994), 26-32.

35. Wang, S. and Kaufman, A., "Volume Sculpting", *ACM Symposium on Interactive 3D Graphics*, Monterey, CA, April 1995, 151-156.

36. Westermann, R., "A Multiresolution Framework for Volume Rendering", *1994 Symposium on Volume Visualization*, Washington, D.C., October 1994, 51-58.

37. Westover, L., "Footprint Evaluation for Volume Rendering", *Computer Graphics (Proc. SIGGRAPH)*, **24**, 4 (August 1990), 144-153.

38. Wright, J. and Hsieh, J., "A Voxel-Based, Forward Projection Algorithm for Rendering Surface and Volumetric Data", *Proc. Visualization '92*, Boston, MA, Oct. 1992, 340-348.

39. Yagel, R., Cohen, D. and Kaufman, A., "Discrete Ray Tracing", *IEEE Computer Graphics & Applications*, **12**, 5 (September 1992), 19-28.

40. Yeo, B. and Liu, B., "Volume Rendering of DCT-Based Compressed 3D Scalar Data", *IEEE Transactions on Visualization and Computer Graphics*, **1**, 1 (March 1995), 29-43.