Visualization Lab

"Volume Graphics"
IEEE Computer, Vol. 26, No. 7
July 1993
pp. 51-64

Volume Graphics

Arie Kaufman, State University of New York at Stony Brook
Daniel Cohen, Ben Gurion University
Roni Yagel, The Ohio State University

1. Introduction

Swift advances in hardware, in particular faster, larger, and cheaper memories, have been transforming revolutionary approaches in computer graphics into reality. One typical example is the revolution of raster graphics that took place in the seventies, when hardware innovations enabled the transition from vector graphics to raster graphics. Another example which has a similar potential is currently shaping up in the field of volume graphics. This trend is rooted in the extensive research and development effort in scientific visualization in general and in volume visualization in particular.

Volume visualization is a method of extracting meaningful information from volumetric datasets through the use of interactive graphics and imaging, and is concerned with the representation, manipulation, and rendering of volumetric datasets [6]. Its objective is to provide mechanisms for peering inside volumetric datasets and for probing into voluminous and complex structures and dynamics. It encompasses an array of techniques for projecting and shading a volumetric dataset or properties thereof and for interactively extracting meaningful information from it using transformations, cuts, segmentation, translucency control, measurements and the like. Typically, the volumetric dataset is represented as a 3D discrete regular grid (i.e., a 3D raster) of volume elements in short, voxels) and is commonly storedin a volume buffer (also called cubic frame buffer), which is a large 3D array of voxels. Alternatively, other data structures and formats have been employed for storing and manipulating the dataset, such as cell decomposition as in octrees [10], sparse voxel matrices, semi-boundaries, voxel runs, irregular grids, and surfaces of objects.

A voxel is the cubic unit of volume centered at the integral grid point. Representing a unit of volume, the voxel is the 3D counterpart of the 2D pixel which represents a unit of area, and thus the volume buffer of voxels can be regarded as the 3D counterpart of the 2D frame buffer of pixels. Each voxel has numeric values associated with it, which represent some measurable properties or independent variables (e.g., color, opacity, density, material, coverage proportion, refractive index, velocity, strength, time) of the real phenomenon o object residing in the unit volume represented by that voxel. The aggregate of voxels tessellating the volume buffer forms the volumetric dataset [6]. (See also Glossary sidebar.)

The source of volume data is sampled data of real objects or phenomena, computed data pro-duced by a computer simulation, or modeled data generated from a geometric model. Examples of applications generating sampled data are medical imaging (e.g., computed tomography, magnetic resonance imaging, ultrasonography), biology (e.g., confocal microscopy), geoscience (e.g., seismic measurements), industry (i.e., industrial CT inspection), and molecular systems (e.g., electron density maps) [6]. Some examples of applications generating computed datasets by typically running a simulation on a supercomputer are meteorology (e.g., storm prediction), computational fluid dynamics (e.g., water flow), and computational chemistry (e.g., new materials).

Although 3D raster representation seems to be more natural for empirical imagery, due to its ability to represent interiors and digital samples, the advantages of this representation are also attracting traditional surface-based applications that deal with the modeling and rendering of synthetic scenes represented by geometric models. Some examples are the rendering of fractals [11], hypertextures [12], fur [5], gases [3], and other complex models [13] including CAD models and terrain models for flight simulators [14]. Furthermore, in many applications involving sampled data, like surgical planning and radiation therapy planning, the data need to be visualized along with synthetic objects that may not be available in digital form, such as prosthetic devices, scalpels, injection needles, isodose surfaces, and radiation beams. The geometric objects can be converted into voxel representation (voxelized) and intermixed with the sampled organ in the voxel buffer [7].

Volume graphics, which is the subject of this paper, is an emerging subfield of computer graphics concerned with the synthesis, manipulation, and rendering of volumetric objects, stored in a volume buffer of voxels. Unlike volume visualization which focuses primarily on sampled and computed datasets, volume graphics is concerned primarily with modeled geometric scenes and particularly with those that are represented in a regular volume buffer (see also the Glossary sidebar). As an approach, volume graphics has the potential to greatly advance the field of 3D graphics by offering a comprehensive alternative to traditional surface graphics.

Figure 1 portrays the taxonomy and the dataflow of volume visualization and volume graphics. In this figure the use of volume graphics techniques in various stages of volume visualization is marked with solid lines. The major sources of volumetric data, displayed at the top, are sampled/computed data (on the left) and geometric models (on the right). The sampled/computed input are 3D reconstructed to fill gaps of missing information and are then a stored in the volume buffer. The geometric model in 3D continuous space is represented by geometric formula which is 3D scan-converted (voxelized) into a set of voxels that "best" approximate the model and is stored in the volume buffer (see [6] Chapter 5). The fundamentals of voxelization and the related 3D discrete topology issues are presented in the Fundamentals of Voxelization sidebar.

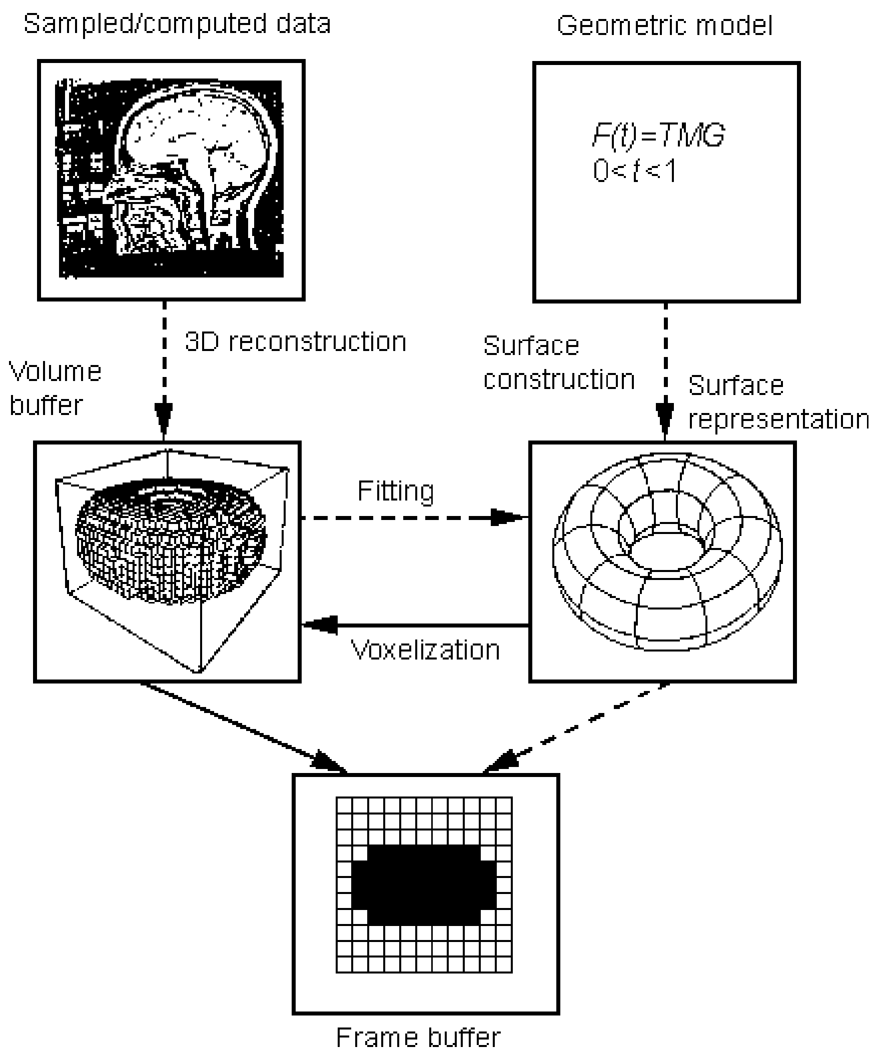

Figure 1: Taxonomy and dataflow of volume visualization and volume graphics. The solid arrows mark the primary volume graphics dataflow.

In order to visualize the dataset in the volume buffer, the sampled/computed data can be con verted into a geometric model by fitting geometric primitives to surfaces that have been a detected in the volume. This set of primitives is then rendered to the screen by employing traditional surface rendering algorithm. However, the preferred alternative for volume viewing is that the volume primitives are directly projected onto a 2D pixel buffer. This process, which is termed volume rendering, involves both the viewing and the shading of the volume image and can be accomplished by forward projection [1], by ray casting [9], or by discrete ray tracing (see the Discrete Ray Tracing sidebar and [6] Chapters 3 and 4).

We now turn to briefly describe, in Section 2, the primary reasons behind the transition in 2D graphics from vector-based to raster-based approach. We then focus, in Section 3, on volume graphics and show that it has the potential to promote a similar transition in 3D graphics as an alternative to traditional surface graphics. Sections 4 and 5 respectively elaborate on the weaknesses and advantages of volume graphics.

2. From Vector Graphics to Raster Graphics

The display of graphics in the sixties and seventies was based on vector drawing devices and on an object-based approach to scene representation, manipulation, and display. A geometric representation of the objects comprising the scene was stored in a display-list. Refreshing the screen was accomplished by redrawing the vectors comprising the objects in the display-list. The major advantages of vector graphics were its ability to perform object related operations on the display-list and the fact that the vectors it drew were continuous and thus exhibited no aliasing. This technology, however, offered calligraphic drawing only, while the interior shaded areas were extremely hard to render

The alternative approach, termed raster graphics, has been predominant since the late seventies. Raster graphics utilizes a 2D frame-buffer (a raster) of pixels for scene representation and a point-based renderer for coloring those pixels that correspond to the discrete representation of the geometric objects. Screen refresh is performed by a video controller, which repeatedly displays the frame-buffer onto the screen [4].

Table 1: A comparison between vector graphics and raster graphics.

Capability	Vector-Graphics	Raster-Graphics
1. Rendering and screen-refresh	Rendering is embedded in screen-refresh	Scan-conversionis decoupled from screen-refresh
2. Rendering performance	Sensitive to scene and object complexity	Insensitive to scene and object complexity
3. Memory and processing requirement	Variable - depends on scene and object complexity	Large but constant
4. Screen space aliasing	None	Frequent
5. Transformation	Continuous, performed on the geometric definition of objects	Discrete, performed on pixel blocks (windows)
6. Boolean and block operations	Difficult, must be performed analytically	Trivial, by employing bitblt, pixel-by-pixel operation, aggregation, quadtrees
7. Rendering of interior	No, boundary only	Yes, colored, shaded and textured surfaces
8. Adequacy for sampled digital images	No	Yes
9. Measurements (e.g., distance, area)	Analytical, but often complex	Discrete approximation , but simple

Table 1 contrasts vector graphics with raster graphics. Unlike vector graphics, raster graphics provides the capability to present realistic, shaded, and textured surfaces in full color, as well as line drawings (row 7 in Table 1). The main disadvantages of this approach are the aliasing present in the image due to the discrete nature of the representation (row 4), and the large memory and processing power this approach requires (row 3). The latter two difficulties delayed the full acceptance of raster graphics until the late seventies when the technology was able to provide cheaper and faster memory and hardware to support the demands of the raster approach. In addition, the discrete nature of rasters makes them less suitable for geometric operations such as transformations (row 5) and accurate measurements (row 9).

On the other hand, a main appeal of raster graphics is that it decouples image generation from screen refresh (row 1), thus making the refresh task insensitive to the scene complexity (row 2). In addition, the raster representation lends itself to block operations, such as bitblt (bit block-transfer), in which a window or a rectangular block of pixels can be rapidly transferred with a variety of pixel-by-pixel operations between the source and destination blocks (row 6) [4]. Raster graphics is also suitable for displaying 2D sampled digital images, and thus provides the ideal environment for mixing digital images with synthetic graphics (row 8). These advantages, coupled with advances in hardware and the development of antialiasing methods, have led raster graphics to replace vector graphics as the primary technology for computer graphics.

3. From Surface Graphics to Volume Graphics

The object-based approach of vector graphics has been adapted for 3D graphics at the expense of maintaining and manipulating a display-list of geometric objects and regenerating the frame-buffer after every change in the scene or viewing parameters. This approach, termed surface graphics, combines raster technology for the display and an object-based approach for the representation, manipulation and rendering of 3D scenes. This method is supported by powerful *geometry engines*, which constitute the present hardware for polygon rendering. These have flourished in the past decade, making surface graphics the state-of-the-art in 3D graphics [4].

Surface graphics strikingly resembles vector graphics in many ways. Like vector graphics surface graphics represents the scene as a set of geometric primitives kept in a display-list. These primitives are transformed, mapped to the screen coordinates, and converted by scan-conversion algorithms into a discrete set of pixels, which is stored in the frame-buffer. This digitization process is also called *rasterization or pixelization.* Any change to the scene, viewing parameters, or shading parameters requires the image generation system to repeat this process and reprocess the complete scene description. Surface graphics generates merely the surfaces of 3D solid objects viewed from a given direction, and subject to limitations similar to those of vector graphics, it does not support the rendering of the interior of these 3D objects.

Instead of a list of geometric objects, volume graphics employs a 3D volume buffer as a medium for the representation and manipulation of 3D scenes. A 3D scene is discretized ear-lier in the image generation pipeline, 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 (viewing and shading). 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 the same benefits as surface graphics, with several advantages that are due to the decoupling, uniformity, and atomicity features. The rendering phase is viewpoint independent and insensitive to scene complexity and object complexity. It supports Boolean and block operations and constructive solid modeling. When 3D sampled and simulated data is available, such as that generated by medical scanners (e.g., CT, MRI) or scientific simulations (e.g., CFD), volume graphic is suitable for their representation. It is capable of representing amorphous phenomena and it can have information on both the interior and exterior of 3D objects. Several disadvantages of this approach are related to the discrete nature of the representation, namely, that transformations and shading are performed in discrete space. In addition, this approach requires substantial amounts of storage space and specialized processing. These disadvantages and advantages as compared with surface graphics are discussed in detail in the following sections.

The same appeal that drove the evolution of the computer graphics world from vector graph ics to raster graphics, once the memory and processing power became available, is starting to drive a variety of applications from surface-based representation of 3D scenes to voxel-based representation. Naturally, this trend first appeared in applications involving sampled 3D data, such as medicine and scientific visualization, in which the datasets are in volumetric form. The diverse empirical imagery applications of volume visualization still provide a major driving force for advances in volume graphics. Table 2 contrasts volume graphics and surface graphics. In Section 4 we discuss the disadvan-tages of volume graphics relative to surface graphics, which are due to its discrete form (rows 3, 4, and 9 in Table 2), the loss of geometric information, and the memory and processing power it requires (row 2). Then, in Section 5 we discuss the advantages of volume graphics: its insensitivity to scene and object complexity (row 1), its viewpoint independence (rows 5 and 10), its ability to represent sampled and simulated datasets (row 8), its ability to represent inner information and amorphous phenomena such as clouds and smoke (row 7), and its ability to support various block operations (row 6).

Table 2: A comparison between Surface graphics and Volume graphics.

Capability	Surface Graphics	Volume Graphics
1. Rendering performance	Sensitive to scene and object complexity	Insensitive to scene and object complexity
2. Memory and processing requirement	Variable - depends on scene and object complexity	Large but constant
3. Object-space aliasing	None	Frequent
4. Transformation	Continuous, performed on the geometric definition of objects	Discrete, performed on pixel blocks (windows)

5. Scan conversion and rendering	Pixelization is embedded in viewing	Voxelization is decoupled from viewing
6. Boolean and block operations	Difficult, must be performed analytically	Trivial, by using voxblt, voxel-by-voxel operation, aggregation, octrees
7. Rendering of interior and amorphous phenomena	No, surface only	Yes, rendering of inner structures as well as surfaces
8. Adequacy for sampled data and intermixing with geometric data	Partially and indirectly (fitting followed by surface rendering)	Supports representation and direct rendering
9. Measurements (e.g., distance, area, volume, normal)	Analytical, but often complex	Discrete approximation , but simple
10. Viewpoint dependency	Requires recalculation for every viewpoint change	Precomputes and stores viewpoint-independent information

4. Disadvantages of Volume Graphics

Discrete Form

Unlike surface graphics, in volume graphics the 3D scene is represented in discrete form. This is the cause of many of the maladies of voxel-based graphics, which are similar to those of 2D rasters [2]. The finite resolution of the raster poses a limit on the accuracy of some operations, such as volume and area measurements, that are based on voxel counting (row 9 in Table 2). Manipulation and transformation of the discrete volume are difficult to achieve without degrading the image quality or losing some information (row 4). Rotation of rasters by angles other than 90 degrees is especially problematic since a sequence of consecutive rotations will distort the image.

Since the continuous object is reconstructed by sampling the discrete data during rendering, a low resolution volume yields high aliasing artifacts (row 3 in Table 2). This becomes espe-cially apparent when zooming in on the 3D raster. When naive rendering algorithms are used, the 3D discrete points may appear to be parted from each other, and may cause the appear-ance of holes. Nevertheless, this can be alleviated to some extent in ways similar to those ,adopted by 2D raster graphics, such as employing either reconstruction techniques (e.g. supersampling, filtering) or a high-resolution volume buffer.

Loss of Geometric Information

In volume graphics we allow each voxel to maintain only local information pertaining to the volume unit it represents. After a surface object has been voxelized, the voxels comprising the discrete object do not retain any geometric information regarding the surface definition of the object. Thus, it is advantageous, when exact measurements are required (e.g., distance, volume, area), to employ surface-based modeling where the geometric surface definition of the object is available. A voxel-based object is only a discrete approximation of the original continuous object where the volume buffer resolution determines the precision of such measurements. On the other hand, several measurement types are more easily computed in voxel space (e.g., mass property, adjacency detection, andvolume computation) (row 9 in Table 2).

The lack of geometric information in the voxel may inflict other difficulties,such as those encountered when rendering discrete surfaces. An essential requirement for most shading methods is the ability to calculate the normal vector to the surfaces comprising the 3D scene. In traditional surface graphics, normal vectors are either analytically calculated from the surface representation or stored as part of the surface representation. In voxel-based models, a discrete shading method is employed to estimate the normal froma context of voxels. A variety of image-based and object-based methods for normal estimation from volumetric data has been devised (see [6] Chapter 4), most methods based on fitting some type of a surface primitive to a small neighborhood of voxels. Nevertheless, this subject is still an active field of research.

A partial integration between surface and volume graphics is conceivable as part of an object based approach in which an auxiliary object table, consisting of the geometric definition and global attributes of each object, is maintained in addition to the volume buffer. Each voxel consists of only an index to the object table, allowing exact calculation of normal, exact measurements, and intersection verification for discrete ray tracing (see Discrete Ray Tracing sidebar) . The auxiliary geometric information might be useful for re-voxelizing the scene in case of a change in the scene itself.

Memory and Processing Memory and Processing.

A typical volume buffer occupies a large amount of memory; for example, for a moderate resolution of 512^3 the volume buffer consists of more than 10^8 voxels. Even if we allocate only one byte per voxel, 128M bytes will be required (row 2 in Table 2). However, since computer memories are significantly decreasing in price and increasing in their compactness and speed, such large memories are becoming more and more feasible. This argument echoes similar discussion when raster graphics emerged as a technology in the mid-seventies. With the rapid progress in memory price and compactness, it is safe to predict that, as in the case of raster graphics, the memory will soon cease to be a stumbling block for volume graphics.

Nevertheless, the extremely large throughput that has to be handled requires a special architecture and processing attention (see [6] Chapter 6). *Volume engines*, analogues to the currently available geometry engines, are emerging. Because of the presortedness of the volume buffer and the fact that only a single type of object - the voxel - has to be handled, volume engines are conceptually simpler to implement than current geometry engines. We predict that, consequently, volume engines will materialize in the near future, with capabilities to synthesize, load, store, manipulate, and render volumetric scenes in real time (e.g., render 30 frames/sec), configured possibly as accelerators or co-systems to existing geometry engines.

5. Advantages of Volume Graphics

Insensitivity to Scene Complexity.

One of the most appealing attributes of volume graphics is its insensitivity to the complexity 1 of the scene, since all objects have been pre-converted into a finite size volume buffer (row in Table 2). Although the performance of the voxelization phase is influenced by the scene complexity, rendering performance depends mainly on the constant resolution of the volume buffer and not on the number of objects in the scene. This is in contrast to representing the volume with an octree whose size varies according to the scene complexity [10]. Insensitivity to the scene complexity makes the volumetric approach especially attractive for scenes consisting of a large number of objects, such as those generated by fractal systems (see Figure 2). Another example of such a scene is a curved surface represented by a large polygon mesh that is generated by a polyhedral smoothing or fitting algorithm. A polygon mesh can approximate a curved surface, where the approximation precision and presentation quality increase with the number of polygons in the mesh. However, using a very fine mesh in conventional surface graphics is expensive with respect to space and display time.

Insensitivity to Object Complexity.

In volume graphics, rendering (viewing and shading) is decoupled from digitization (voxelization) and all objects are first converted into one meta object, the voxel, which makes the rendering process insensitive to the complexity of the objects (row 5 in Table 2). Thus, volume graphics is particularly attractive for objects that are hard to render using conventional graphics systems. Examples of such objects include curved surfaces of high order and fractals which require the expensive computation of an iterative function for each volume unit [11] (see also Figure 2). Constructive solid models are also hard to render by conventional methods but are straightforward to render in volumetric representation (see a separate discussion below in Block Operations and Figures 2-6 and 9).

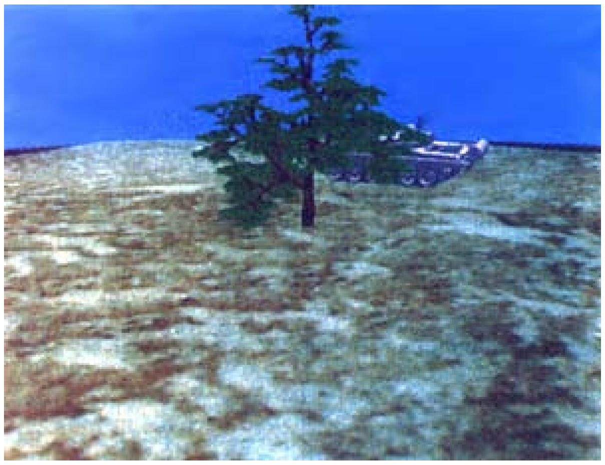

Figure 2: A T62 tank made of about 40,000 voxelized polygons, partially occluded by a tree made of 20,000, fractally grown, voxelized polygons. The tank resolution is about 1.5 inches per voxel in the foreground, which translates to about a 256^3 voxel model.

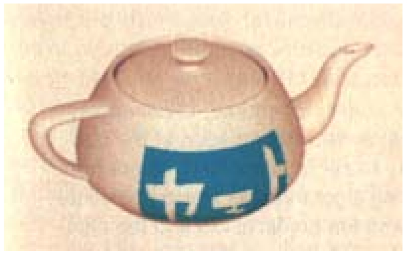

Figure 3: A teapot modeled by 32 Bezier patches and voxelized to a 512^3 resolution voxel model. The generated voxels were assigned color during the voxelization process according to a 2D texture map, while the shading has been applied during the rendering process. voxel model.

Another type of object complexity involves objects that are enhanced with a technique known as *texture-mapping*, where the realism of objects is increased by simulating surface details. Texture-mapping is commonly implemented during the last stage of the rendering pipeline where the texture is extracted from a 2D texture image and mapped onto the surface to be rendered, and its complexity is proportional to the object complexity [4]. In volume graphics, texture-mapping is performed only once, during the voxelization stage, where the texture color is calculated and stored in each voxel. Solid texturing, which employs a 3D texture image, has also a high complexity similar to texture-mapping [4]. In volume graphics, however, solid texturing, like texture-mapping, is performed during the voxelization stage.

The textured object in Figures 2, 3 and 4 has been assigned texture during the voxelization stage by mapping each voxel back to the corresponding value on a texture surrounding the object. Figures 10, 11, and 12 show voxelized terrain that has been mapped with satellite or aerial photos during the voxelization stage.

We have also implemented a photo-mapping technique where six photographs of the real object are projected back onto the voxelized object (see Figure 6 and the building in Figure 12). Once this mapping is applied, it is stored with the voxels themselves during the voxeli-zation stage, which does not degrade the rendering performance. In addition, texture-mapping and photo-mapping are also viewpoint independent attributes, implying that once the texture is stored as part of the voxel value, texture-mapping need not be repeated.

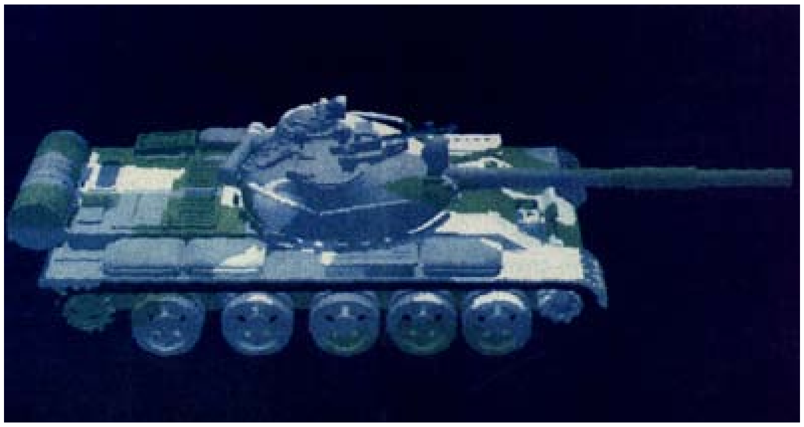

Figure 4: A camouflaged T62 tank voxelized from a 50,000 polygon model into a 2 inch per voxel resolution. Voxels were given color during the voxelization process according to a spherical texture map of pseudo-camouflage. The tread is invisible because it blends with the background except over the back sprocket.

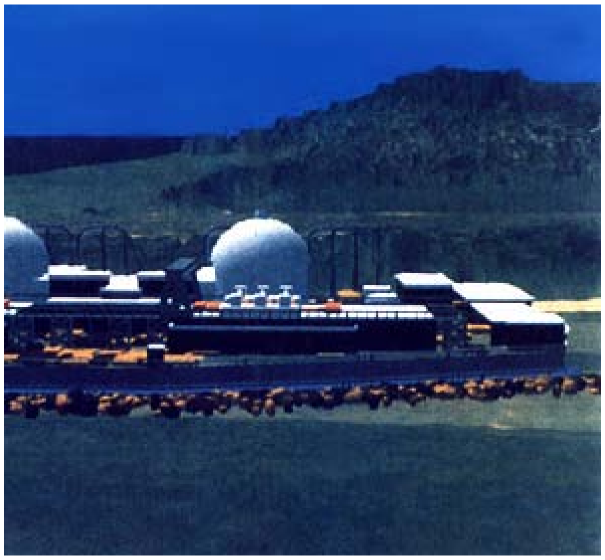

Figure 5: San Onofre Nuclear Power Plant embedded in a terrain database of upstate New York. This image was voxelized from a plant model consisting of about 45,000 polygons into 2000 X 700 voxels horizontally with a one foot per voxel resolution. The terrain resolution is about six inches per voxel.

Viewpoint Independence.

A main difference between voxel-based graphics and conventional surface graphics is that in the former the scene is discretized (voxelized) once for multiple viewing conditions, while in the latter the scene is repeatedly scan-converted after every change in the viewing parameters, causing a performance bottleneck in its rendering pipeline (row 10 in Table 2). This attractive advantage of volume graphics can be attributed in part to the fact that, in the volumetric representation, a unit of memory is allocated for each unit of space, in contrast to surface graphics, where memory is assigned only to complete surface patches. This enables volume graphics to store view independent attributes at each volume unit, while surface graphics is not able to provide storage for attributes that vary across its basic surface elements.

In anticipation of repeated access to the volume buffer (such as in animation), all viewpoint independent attributes can be precomputed during the voxelization stage, stored with the voxel, and be readily accessible for speeding up the rendering. The voxelization algorithm can generate for each object voxel its color, its texture color, its normal vector (for visible voxels), and information concerning the visibility of the light sources from that voxel. Actually, the viewpoint independent parts of the illumination equation, that is, the ambient, illumination and the sum of the attenuated diffuse illumination of all the visible light sources [4] can also be precomputed and stored as part of the voxel value.

Once a volume buffer with precomputed view-independent attributes is available, a rendering algorithm such as a discrete ray tracing algorithm can be engaged. Discrete ray tracing is based on traversing 3D discrete rays through the volume buffer, and is described briefly in the sidebar on Discrete Ray Tracing. The discrete ray tracing approach is especially attractive for ray tracing complex surface scenes and constructive solid models, as well as 3D sampled and computed datasets (see below). Figures 7 and 8 show examples of objects that were voxelized and then ray traced in discrete voxel space. In spite of the complexity of these scenes, ray tracing time was approximately the same as for much simpler scenes and significantly superior to traditional space-subdivision ray tracing methods. Moreover, in spite of the discrete nature of the volume buffer representation, images indistinguishable from the ones produced by conventional surface-based ray tracing can be generated by employing auxiliary object tables and screen supersampling techniques, which casts several rays per pixel (see Figure 8 and the Discrete Ray Tracing sidebar).

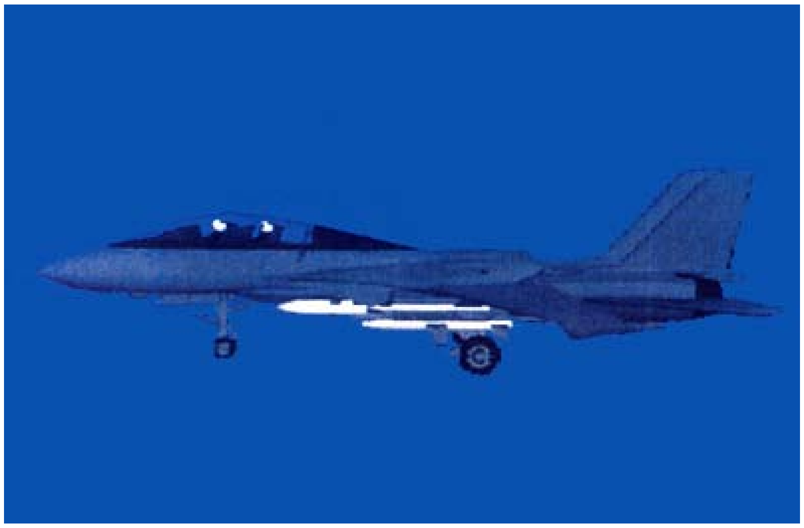

Figure 6: A Tomcat aircraft voxelized in 3 inch per voxel resolution. Note the pilots' helmets visible through the translucent canopy.

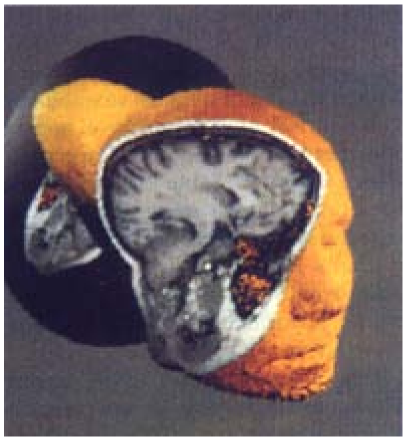

Figure 7: A 256^3 resolution volume of a reconstructed MRI head reflected in a voxelized mirror. The cut was performed by a constructive solid modeling subtraction (voxel-by-voxel) during the voxelization stage.

Sampled and Simulated Datasets.

Sampled datasets (such as in 3D medical imaging, see Figure 7) and simulated datasets (such as in computational fluid dynamics) are often reconstructed from the acquired sampled or simulated points into a regular grid of voxels and stored in a volume buffer. Such datasets provide for the majority of applications using the volumetric approach. Unlike surface graphics, volume graphics naturally and directly supports the representation, manipulation, and rendering of such datasets (row 7 in Table 2), as well as provides the volume buffer medium for intermixing sampled or simulated datasets with geometric objects (row 8) [7], as can be seen in Figures 5, 7, and 9.

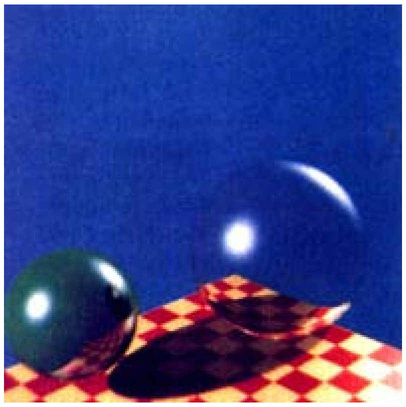

Figure 8: Turner Whitted's spheres and a checkerboard floor voxelized into a 320 volume and ray traced in 377 seconds on a 20 MIPS machine

with geometric intersection verification and supersampling by casting 4 rays per pixel (see the Discrete Ray Tracing sidebar).

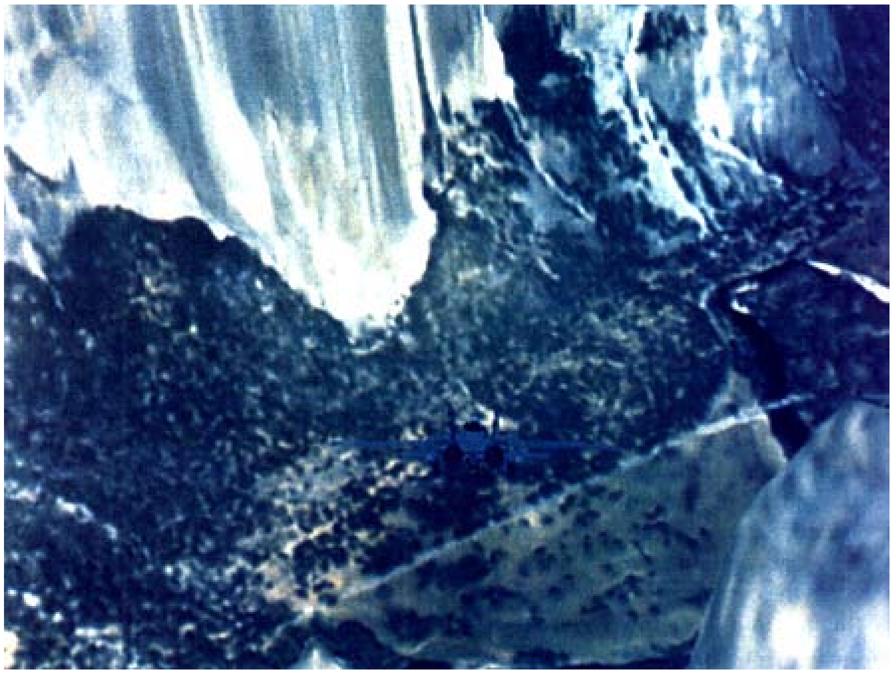
Figure 9: A voxelized Tomcat (see Figure 6) flying over Yosemite Valley (see Figure 11) in a flight simulation of a dog fight.

Inner Information.

A central feature of volumetric representation is that, unlike surface representation, it is capable of representing inner structures of the objects, which can be revealed and explored with the appropriate manipulation and rendering techniques (row 7 in Table 2). Natural objects as well as synthetic objects are likely to be solid rather than hollow. The inner structure is thus an important aspect of image complexity, which is easily explored using volum graphics and cannot be supported by surface graphics (see Figure 7). Moreover, 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 do not have any notion of tangible surfaces [3, 5, 12]. Figure 10 exemplifies the rendition of haze as part of a voxel-based terrain model.

Block Operations.

An intrinsic characteristic of rasters 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 (bit block-transfer) operations, or its 3D counterpart, a voxblt (voxel block-transfer) operations, which support transfer of cuboidal voxel blocks with variety of voxel-by-voxel operations between source and destination blocks (row 6 in Table 2) [8]. Such block operations add a variety of modeling capabilities which aid in the task of image synthesis. Moreover, the volume buffer lends itself to Boolean operations that can be performed on a voxel-by-voxel basis during the voxelization stage. This property is very advantageous when Constructive Solid Geometry (CSG) is the modeling paradigm. CSG operations such as subtraction, union, and intersection between two voxelized objects are accomplished at the voxel level [10], thereby reducing the original problem of evaluating a CSG tree of such operations during rendering time down to a 1D Boolean operation between pairs of voxels during a preprocessing stage. Once a CSG model has been constructed in voxel representation, it is rendered like any other volume buffer. This makes discrete ray tracing of constructive solid models straightforward. Figure 7 shows a volumetric ray tracing of a 256^3 reconstructed MRI head with a CSG subtraction and a back mirror that has been generated by a polygon voxelization algorithm (see Fundamentals of Voxelization sidebar).

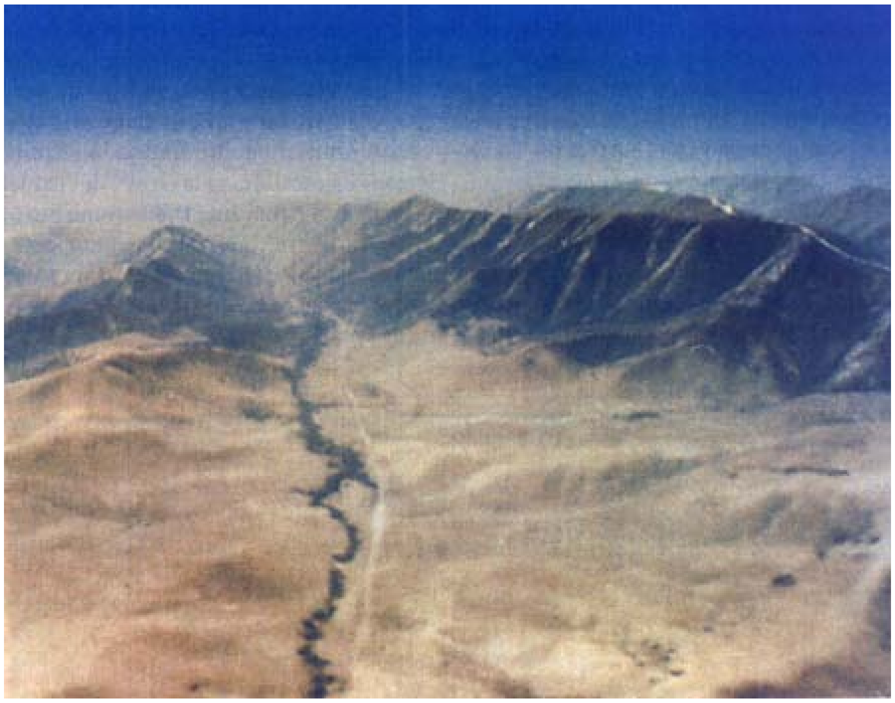

Figure 10: View west over Camp Pendleton in California with moderately dense haze. Resolution is about 16 feet per voxel.

The spatial presortedness of the volume buffer voxels lends itself to other types of grouping or aggregation of neighboring voxels. For example, the terrain images shown in Figures 9 - 11 and in Figure 12 were generated, respectively, by the Hughes Aircraft *RealScene*® flight simulator [14], and by Tiltan System Engineering *SceneGenerator*® . Both systems simulate flight over voxel-represented terrain enhanced with satellite photo-mapping with additional synthetic raised objects, such as buildings, trees, vehicles, aircrafts, clouds and the like (see Figures 5 and 12). Since in this application the information below the terrain surface is invisible, terrain voxels can be represented as tall cuboids extending from sea level to the terrain height. This representation saves both storage space, and retrieval and processing time. The raised objects, however, have to be represented in a conventional voxel-based form.

Similarly, voxels can be aggregated into super-voxels in a pyramid-like hierarchy. For example, in a voxel-based flight simulator, voxels representing 1' X 1' X 1' voxel size can be used for takeoff and landing. As the aircraft ascends, fewer and fewer details need to be processed and visualized, and a 16' X 16' X 16' or larger voxel size will suffice. A hierarchicalvolume buffer can be prepared in advance or on-the-fly by subsampling or averaging the appropriate size neighborhoods of voxels.

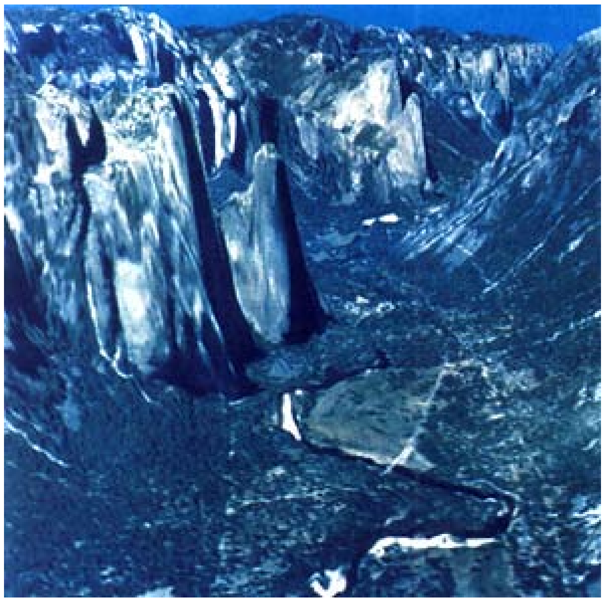

Figure 11: A typical view of a voxelized terrain model in flight simulation. This image is a view down Yosemite Valley showing Cathedral Rocks, where each voxel represent 8 cubic feet.

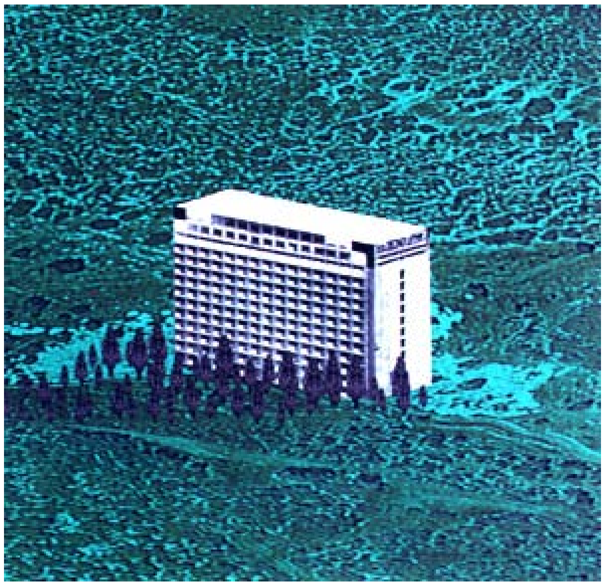

Figure 12: A typical view of a voxelized terrain with an embedded voxelized hotel building and voxelized trees.

6. Conclusions

We have explored volume graphics, which employs a volume buffer for 3D scene representa-tion. As summarized in Table 2, volume graphics has advantages over surface graphics by being viewpoint independent, insensitive to scene and object complexity, and suitable for the representation of sampled and simulated datasets and mixtures thereof with geometric objects. It supports the visualization of internal structures, and lends itself to the realization of block operations, CSG modeling, irregular voxel sizes, and hierarchical representation. The problems associated with the volume buffer representation, such as discreteness, memory size, processing time, and loss 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 computer hardware, and memory systems, coupled with the desire to reveal the inner structures of volumetric objects, suggests that volume graphics will develop into a major trend in computer graphics. Furthermore, the striking simi-larity between Table 1 and Table 2 implies that volume graphics, like raster graphics, has the potential to revolutionize computer graphics. Just as raster graphics in the seventies super-seded 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 scenes composed of surfaces.

Acknowledgments

This work has been supported by the National Science Foundation under grants IRI-9008109, CCR-9205047, and CCR-9211288, and grants from Hughes Aircraft Company and Hewlett Packard. Figure 12 is courtesy of Tiltan System Engineering. Figures 2, 4, 5, 6, 9, 10, and 11 are courtesy of Hughes Aircraft Company. These images have been voxelized using voxelization algorithms, a voxel-based modeler, and a photo-mapper developed at Stony Brook.

7 References

1. Drebin, R. A., Carpenter, L. and Hanrahan, P., "Volume Rendering", Computer Graphics (Proc. SIGGRAPH), 22, 4 (August 1988), 65-74.
2. Eastman, C. M., "Vector versus Raster: A Functional Comarison of Drawing Technologies", IEEE Computer Graphics & Applications, 10, 5 (September 1990), 68-80.
3. Ebert, D. S. and Parent, R. E., "Rendering and Animation of Gaseous Phenomena by Combining Fast Volume and Scanline A-buffer Techniques", Computer Graphics, 24, (August 1990), 367-376.
4. Foley, J., van Dam, A., Feiner, S. and Hughes, J., Computer Graphics: Principles and Practice, Addison-Wesley (Second Edition), 1990.
5. Kajiya, J. T. and Kay, T. L., "Rendering Fur with Three Dimensional Textures", Computer Graphics, 23, 3 (July 1989), 271-280.
6. Kaufman, A., Volume Visualization, IEEE Computer Society Press Tutorial, Los Alamitos, CA, 1990.
7. 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.
8. 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.
9. Levoy, M., "Display of Surfaces from Volume Data", IEEE Computer Graphics and Applications, 8, 5 (May 1988), 29-37.
10. Meagher, D. J., "Geometric Modeling Using Octree Encoding", Computer Graphics and Image Processing, 19, 2 (June 1982), 129-147.
11. Norton, V. A., "Generation and Rendering of Geometric Fractals in 3-D", Computer Graphics, 16, 3 (1982), 61-67.
12. Perlin, K. and Hoffert, E. M., "Hypertexture", Computer Graphics, 23, 3 (July 1989), 253-262.
13. Snyder, J. M. and Barr, A. H., "Ray Tracing Complex Models Containing Surface Tessellations", Computer Graphics, 21, 4 (July 1987), 119-128.
14. Wright, J. and Hsieh, J., "A Voxel-Based, Forward Projection Algorithm for Rendering Surface and Volumetric Data", Proceedings Visualization '92, Boston, MA, October 1992, 340-348.

Sidebars

- **Discrete Ray Tracing**
- **Fundamentals of Voxelization**
- **Glossary**