SCANNING VOL. 22, 218–226 (2000)

Received: January 10, 2000
Accepted: April 20, 2000

Three-Dimensional Microscopy Data Exploration by Interactive Volume Visualization

SHIAOFEN FANG, YI DAI, FREDERICK MYERS, MIHRAN TUCERYAN, KENNETH DUNN*

Department of Computer and Information Science, Indiana University Purdue University Indianapolis; *Department of Medicine, Division of Nephrology, School of Medicine, Indiana University, Indianapolis, Indiana, USA

Summary: This paper presents a new volume visualization approach for three-dimensional (3-D) interactive microscopy data exploration. Because of their unique image characteristics, 3-D microscopy data are often not able to be visualized effectively by conventional volume visualization techniques. In our approach, microscopy visualization is carried out in an interactive data exploration environment, based on a combination of interactive volume rendering techniques and image-based transfer function design methods. Interactive volume rendering is achieved by using two-dimensional (2-D) texture mapping in a Shear-Warp volume rendering algorithm. Image processing techniques are employed and integrated into the rendering pipeline for the definition and searching of appropriate transfer functions that best reflect the user's visualization intentions. These techniques have been implemented successfully in a prototype visualization system on low-end and middle-range SGI desktop workstations. Since only 2-D texture mapping is required, the system can also be easily ported to PC platforms.

Key words: three-dimensional microscopy, visualization, volume rendering, texture mapping, image processing

PACS: 07.05 Rm, 07.05.Pj

Introduction

In the past 20 years, microscopy has come to play an increasingly important role in the study of cell biology. The advances in biochemistry and molecular biology have generated an increased appreciation of the cellular organization of the biochemical components of a cell. Although advances in confocal microscopy and image deconvolution have made it feasible to collect high-resolution (Dunn *et al.* 1994, Shaw 1995), three-dimensional (3-D) image volumes of thick samples such as epithelial cells, application of this technology to 3-D imaging is still in its infancy. Indeed, the proliferation of 3-D microscopy in cell biology has generated vast amounts of image data that have not been sufficiently explored and analyzed.

The rapid advances in computer graphics and visualization, in particular volume visualization, have provided a great potential for the visualization of 3-D microscopy images. Volume visualization is a new 3-D computer graphics technique that is concerned with the abstraction, interpretation, rendering, and manipulation of large volume datasets. Volume-rendering algorithms, for instance, can directly display the entire volume dataset through semi-transparent images and allow the viewer to peer inside the internal structures of the image volume for truly 3-D data viewing and analysis. Although volume visualization methods and tools have been used in many scientific and medical applications, such as visual simulations and computed tomography/magnetic resonance imaging (CT/MRI), its applications in 3-D microscopy are limited and largely ineffective. This is mainly because current volume visualization techniques are mostly designed for CT/MRI types of images and are poorly suited for 3-D microscopy applications. Several unique characteristics of microscopy data pose serious challenges to conventional visualization techniques.

First, fluorescently labeled samples characteristically have low signal levels, sometimes consisting of a single photon, so that microscopy images are typically much noisier than CT or MRI images. Furthermore, since excitation of fluorescence also destroys fluorophores through photobleaching, the signal-to-noise ratio decreases with the collection of each focal plane of an image volume. Consequently, microscopy image volumes are usually very sensitive to small changes in rendering parameters, such as the rendering transfer functions which map image intensity values to colors, opacities, or shading parameters. Thus, ordinary volume visualization algorithms frequently fail to capture the delicate structures present in many cellular objects. Second, structures in the microscopic scale typically show higher complexity than those of the anatomic organs in CT or MRI images. This is particularly true in

Address for reprints:

Shiaofen Fang
Department of Computer and Information Science
Indiana University Purdue University Indianapolis
723 West Michigan Street, SL 280
Indianapolis, IN 46202, USA
email: sfang@cs.iupui.edu

multiparameter images, in which several different proteins will be imaged simultaneously, each in a specific color of fluorescence. A third problem is that the structures of the objects to be examined are often partially or entirely unknown. This leads to the strong need for interactive navigation and searching capabilities in both the spatial dimensions and the transfer function space.

Due to these special characteristics, 3-D microscopy visualization is best performed in a data exploration environment in which users can interactively manipulate, search, and render 3-D microscopy images for their individual visualization goals. Two key technical requirements for such a visualization environment are interactivity and transfer function design.

Data exploration is intrinsically a continuous and interactive process. Although many surface rendering and volume rendering algorithms have been developed (Lacroute and Levoy 1994; Levoy 1988, 1990; Lorensen and Cline 1987; Upson and Keeler 1988; Westover 1990), they have not been able to provide interactive rendering speed to support interactive data exploration. In fact, interactive volume visualization can currently only be obtained using either supercomputers (Parker *et al.* 1988) or special hardware systems, which are not available on common desktop computers. The most popular hardware solution is using 3-D texture mapping hardware which is only available on high-end graphics workstations such as SGI's Onyx systems (Mountain View, Calif.) (Cabral *et al.* 1994, SGI Technical Publications 1998). A recently released hardware chip, *VolumePro*, by Mitsubishi (Irvine, Calif.) also provides real-time volume rendering, but with considerable additional hardware cost. Furthermore, *VolumePro* does not support perspective projection, which is essential in data exploration applications. These limitations severely restrict the usability of volume visualization in 3-D microscopy. Thus, our first goal is to develop a low-cost technique for interactive volume rendering that uses only existing hardware features on common desktop computers.

Another important component of data exploration is the searching for the right transfer functions that best reflect the users' visualization intentions. The transfer function design problem is particularly important and difficult with noisy and unfamiliar data sets, but has not received sufficient research attention. Most current visualization systems employ a trial-and-error approach, which is extremely difficult and time consuming for microscopy data. More important, visualization results obtained this way depend largely on the user's experience and "luck," and can lead to confusing, misleading, and dubious data interpretations. A previous effort in improving the transfer function searching is rather limited. One approach is the evolution-based inverse design approach (He *et al.* 1996, Marks *et al.* 1997), which uses a stochastic search technique to generate many image samples based on an initial population of predefined transfer functions, and then improves the samples based on the user's selections of the sample images at each evolution step. Although this approach provides some level of heuristics for transfer function searching, it is still a very time-consuming process and does not support complicated or procedural transfer functions that cannot be represented by the predefined function combinations. Another related work (Kindlmann and Durkin 1998) uses gradient-based edge detection methods to render volumes in which regions of interest are the boundaries between different materials. The image-based transfer function design approach in this paper is based on a more systematic use of 3-D image processing techniques (Fang *et al.* 1998) of which the approach in Kindlmann and Durkin (1998) is a special case. In our approach, image processing procedures are integrated into the visualization pipeline so that the users can interactively adjust the parameters of the image processing operators for desired and predictable results.

Methods

Interactive 3-D microscopy data exploration can be achieved through a combination of interactive volume rendering and intuitive transfer function design. Technical details of this approach will be given in this section. These include a new interactive volume rendering algorithm using two-dimensional (2-D) texture mapping and a transfer function design method based on image processing operations.

Interactive Volume Rendering by Two-Dimensional Texture Mapping

This algorithm applies 2-D texture mapping in a shear-warp based volume rendering process to achieve interactive speed. There are two important advantages in using 2-D texture mapping over other software- and hardware-based methods. First, 2-D texture mapping is normally implemented in hardware, and therefore is faster than equivalent operations using CPU computations. Second, unlike 3-D texture mapping hardware that is only available on selected high-end graphics workstations, hardware-implemented 2-D texture mapping is widely available and is usually a standard feature on most desktop workstations and personal computers. Combining 2-D texture mapping and a shear-warp factorization technique, we are able to achieve interactive volume rendering without special hardware requirements.

Volume rendering using shear-warp factorization was first proposed in Cameron and Undrill (1992) and later optimized in Lacroute and Levoy (1994). Although the algorithm given in Lacroute and Levoy (1994) is one of the fastest, it still does not provide interactive rendering performance. More important, the algorithm carries two limitations that make it unsuitable for data exploration applications: (1) the algorithm slows down considerably when using perspective projection, and (2) it requires an expensive preprocessing step for data classification with every change of the transfer function. Unfortunately, both the per-

spective viewing and the continuous change of transfer function are crucial features in data exploration.

Shear-warp algorithm is based on the shear-warp factorization of the viewing matrix:

$$M = P \cdot S \cdot W$$

where P is a permutation matrix that transposes the coordinate system to allow the z-axis to be the principal viewing axis, S is a shearing transformation, and W is a warping matrix that is computed by $W = S^{-1} \cdot P^{-1} \cdot M$. The basic steps of the shear-warp algorithm are the following: the volume data set is first transformed to a sheared object space by translating, scaling, and resampling the slices of the volume; these slices are then composited together in a front-to-back order, which essentially projects the slices onto an intermediate image in the sheared object space. A warping operation is finally applied to this intermediate image to generate the correct image using the warping matrix W. This process is illustrated (upper branch) in Figure 1.

The main computational cost in this process is the resampling of the slices in the sheared object space and the subsequent composition of the slices. For perspective viewing, this can be particularly expensive since all slices are scaled differently and thus need to be resampled differently. Our approach considers a slice to have two components: the image component, which is the 2-D image of the slice from the original data set, and the polygon component, which represents the rectangular geometry of the slice. Every time the polygon is geometrically transformed (e.g., translation and scaling) an image resampling needs to be done for the rasterization of the slice in the frame buffer. This process can, however, be accelerated by graphics hardware if the image of each slice is separately defined as a 2-D texture and mapped to its polygon when it is drawn to the frame buffer by the graphics subsystem. Since the texture mapping process involves the resampling computation (by hardware), this is a much faster operation than a CPU-only solution, as shown in Figure 1.

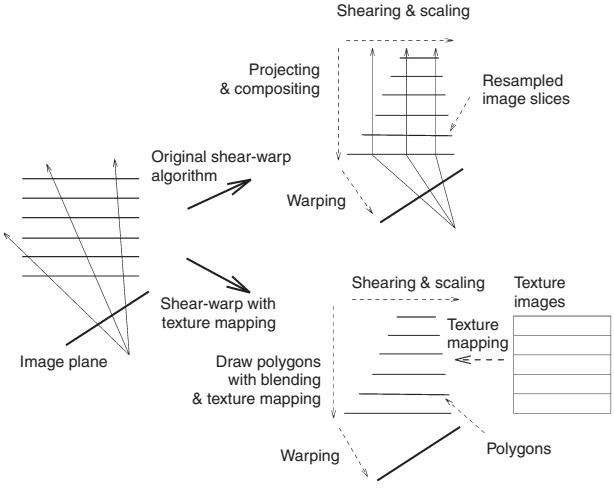

FIG. 1 Shear-warp algorithm: Shear-warp volume rendering by resampling and by 2-D texture mapping.

As in the original shear-warp algorithm (Lacroute and Levoy 1994), three sets of the slices of the volume need to be defined for the three different major viewing axes. The polygon of each slice can be generated on-the-fly during rendering, but its texture image needs to be predefined and stored in the system for fast texture mapping. Since both parallel and perspective viewings of polygons are handled automatically by the graphics subsystem, there is virtually no speed difference between parallel and perspective projections. Unlike the algorithm given in Lacroute and Levoy (1994), where special data structures (e.g., run-length encoding) need to be reconstructed every time the transfer function is modified, the new algorithm extracts the texture images directly from the original data set, independent of the transfer functions, and therefore does not require extra preprocessing when editing the transfer functions. Finally, the warping step can also be conveniently carried out by 2-D texture mapping.

This algorithm, however, has two drawbacks.

Memory requirements: Since no data compression is employed for the three sets of texture images, memory requirements are large. For instance, a 256^3 volume would require over 48 MB memory. A similar amount of memory is required for typical microscopy data sets of size $512 \times 512 \times 64$. However, since memory price has been dropping at a faster pace than other hardware components, this may not be a major concern for most users.

Lack of shading: Two-dimensional texture mapping can not efficiently support shading that displays more realistic surface features. Thus, our algorithm does not generate shaded images, which may result in a loss of quality for surface-rich data sets. However, this does not appear to be a major problem with microscopy data sets in which surfaces are normally not well defined due to the nature of flouresscently labeled samples.

Image-Based Transfer Function Design

The need for transfer function design comes from the dynamic and often subjective visualization goals and requirements in microscopy data exploration, where the users need to search and manipulate the transfer functions interactively in the visualization process to view different types of substructures, surfaces, and frequencies with different visual attributes. Thus, an intuitive and efficient transfer function model is essential for this type of problem.

Transfer functions: A transfer function is a function or a procedure defined over the intensity-spatial domain of a volume data set. It computes a new intensity value for each sample point in the volume space during rendering. It can also be applied to the voxel points of the entire volume to reconstruct a new volume. The intensity values generated from a transfer function can be further mapped to color and opacity values using a color look-up table, rep-

resenting a piecewise linear mapping for each color/opacity component. We call this step the coloring step. Since an intensity volume is essentially a 3-D intensity image, a transfer function can be naturally considered to be an image-processing problem. In this model, a transfer function, F: $\nu \rightarrow \nu$, is defined as a sequence of mappings:

$$F = f_n \circ f_{n-1} \circ \cdots \circ f_2 \circ f_1$$

where ν is the volume data space, and f_i: $\nu \rightarrow \nu$ correspond to a sequence of image processing procedures with adjustable parameters. This sequence and its parameter set uniquely define one transfer function in the transfer function space. For computational simplicity, f_i is restricted to be one of the following two types of mappings:

1. *Intensity table*. It is an intensity-to-intensity look-up table representing a piecewise linear function over the volume's intensity field.
2. *Neighborhood function*. It is a function computed from the intensity values in an $m \times m \times m$ neighborhood of a given voxel, where the neighborhood size, m, is an adjustable parameter of the transfer function. A median filter (Rosenfeld and Kak 1982), for instance, can be considered as a neighborhood function. A more typical example is the 3-D spatial convolution, as a 3-D linear filter of a volume V with an $m \times m \times m$ mask T:

$$f(x,y,z) = \sum_{i,j,k=-\frac{m}{2}}^{\frac{m}{2}} T[i,j,k] \cdot V[x+i, y+j, z+k]$$

Some higher order image processing operations, such as dilation/erosion and anisotropic diffusion (Perona and Malik 1990), cannot be directly represented as a neighborhood function; but these operations are normally applied in some precomputation processes for the definitions of simpler functions.

Integrating transfer functions into rendering: Volume rendering using this transfer function model requires the integration of image processing procedures into the rendering pipeline. This enables the interactive data exploration through transfer function manipulation. Different integration approaches need to be used with different types of volume rendering algorithms. These include point-based approach, volume-based approach, and slice-based approach (Fig. 2).

Point based approach: This approach can be applied when point is the basic data element processed through the rendering pipeline. Raycasting algorithms are the most typical of this kind. Essentially, every access to an intensity value during rendering will directly go through the computations of all the image processing procedures, as shown in Figure 2a. The main advantage of this approach is that it only computes the image processing operations on points that are actually used by the rendering algorithm, that is, the visible points. Since the number of sample points used for rendering is normally much smaller (<10% in our experience) than the total number of voxels in the volume, this approach is more efficient than applying the transfer function to the entire volume, particularly when frequent changes of the transfer functions are necessary.

Applying intensity tables to points is very straightforward using color and opacity look-up tables, often in hardware (e.g., in OpenGL). The integration of neighborhood functions are, however, more costly. A straightforward approach is to make recursive procedural calls to the neighborhood functions to compute the image processing results dynamically for individual sample points when they are accessed by the rendering algorithm. One problem with this

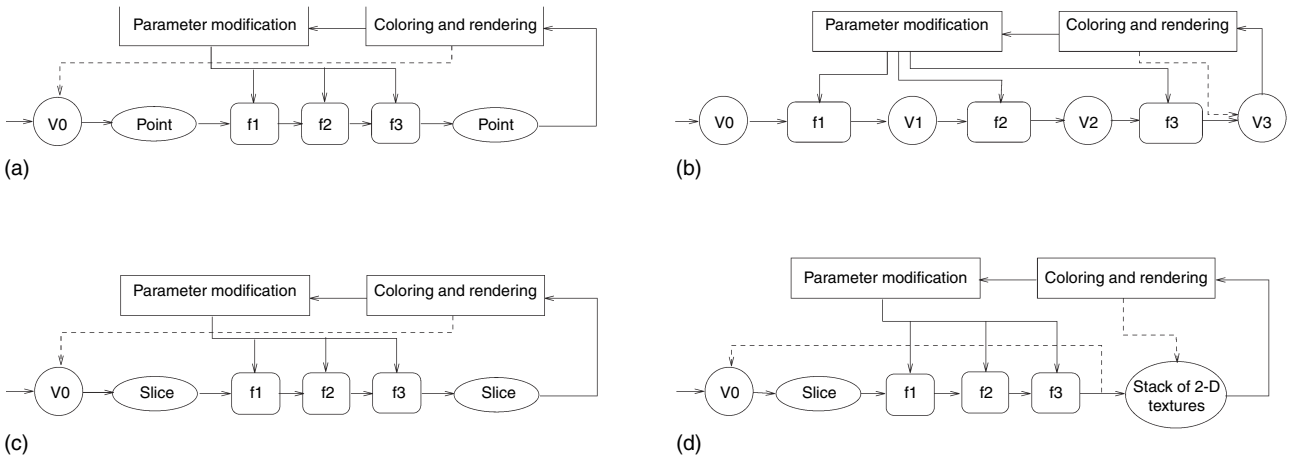

FIG. 2 Integrating image processing into the rendering pipeline: (a) Point-based approach, (b) volume-based approach, (c) slice-based approach, (d) 2-D texture-based approach.

process, however, is the potentially repeated computation with multiple neighborhood functions. Since each voxel can fall into the neighborhoods of several other voxels, it may therefore be accessed (and computed) multiple times when more than one neighborhood functions exist in a transfer function. When the number of neighborhood functions in a sequence is large, such overhead can be significant. Fortunately, the problem can be alleviated by applying some buffering mechanism to store the intermediate results of each computed sample point for possible repeat accesses. For instance, a buffer can be used for each neighborhood function to store the results of all sample points going through this function. This way, the overhead can be partially or entirely eliminated.

Volume-based approach: In this approach, the neighborhood function procedures in the transfer function are applied to the entire volume to generate a sequence of intermediate volumes, with the final volume passed to the coloring step, as shown in Figure 2b. This approach is suitable for rendering algorithms that do not have total control and access to individual voxels of the data set (or are inefficient to do so). The most typical case is the volume-rendering approach using 3-D texture mapping. When the 3-D texture mapping is implemented in hardware, the transfer function computation cannot be easily inserted into the texture mapping pipeline. In this case, applying the transfer function at the volume level is more practical and efficient. Since 3-D texture mapping-based algorithms do not normally have early termination control (Levoy 1990) for avoiding computing points that no longer contribute to the image, a point-based approach does not really have a clear advantage in this case. The implementation of this approach is also simpler since the image processing and volume rendering are decoupled. For instance, special rendering hardware, such as the *VolumePro* chip by Mitsubishi may be used following the transfer function computation.

Slice-based approach: This approach applies the transfer function to one slice of the volume at a time, as shown in Figure 2c; it is suitable for algorithms that access the volume in a slice-by-slice order. Our 2-D texture mapping-based shear-warp algorithm belongs to this category. Some 3-D texture mapping-based algorithms can also be classified into this category. For instance, in the algorithm given in Cabral *et al.* (1994) images are formed by the texture mapping and blending of Z-planes, which are essentially resampled 2-D slices that are perpendicular to the viewing direction. Compared with the volume-based approach, the slice-based approach uses less memory and is also more efficient and flexible when the algorithm needs dynamic determination of the sampling resolutions of the slices and the number of slices to be sampled based on runtime conditions; that is, only the samples that are actually used by the rendering procedure are computed by the transfer function.

In practice, once a transfer function is defined, its rendering result often needs to be examined from many different angles before a change is made to the transfer function. Thus, it is important to be able to reuse the transfer function results for a sequence of renderings. This can be done naturally with our 2-D texture mapping-based shear-warp algorithm. Essentially, each slice is computed by the transfer function before it is saved as a 2-D texture. Thus, the rendering process only communicates with the 2-D texture images generated from the slices, and each change of the transfer function requires a reloading of the 2-D texture images. This process is shown in Figure 2d. Hardware acceleration is also possible for image processing computation. In our implementation, the *ImageVision* system on SGI workstations is used to compute chains of image processing operations. Since *ImageVision* takes advantage of the system's graphics hardware whenever possible, it is usually faster than pure software implementation. Other real-time image processing solutions are also available such as the DSP-based image processing chips by Texas Instruments (Dallas, Tex.).

Image enhancement for data filtering: In this image-based transfer function model, we mainly focus on two types of operations: image enhancement and boundary detection. The goal of image enhancement is to improve the quality of the 3-D image volume for better visual appearance, based on the user's visualization goals; that is, it is basically a data filtering mechanism. Two types of image enhancement techniques are commonly used: point enhancement and spatial enhancement.

A point enhancement operation applies some function to each intensity value individually to generate a new value. Since the result of a point enhancement operation only depends on the intensity value of the point to which it is applied, the corresponding intensity mapping can be represented as an intensity table in our transfer function model. The most common point enhancement operation is intensity modifications, in which the intensity curve of the input volume is altered in certain intensity intervals to increase or reduce the exposure of the corresponding regions. Similar modification scan also be applied to the histogram curve (e.g., histogram equalization). Although the parameters of such operations (e.g., the intensity intervals) can be defined and adjusted directly by the users, they more likely will come from the output of some other image processing procedures, such as boundary detections, in a visualization process.

A spatial enhancement operation derives the new intensity value of a given point from its neighborhood points, that is, the result is neighborhood dependent. Therefore, spatial enhancement operations can only be represented as neighborhood functions in our transfer function model. In general, spatial operations can be classified into smoothing and sharpening operations.

Smoothing operations are primarily used to remove image noise. We sometimes also want to remove very small feature details for better presentation of the larger features. One example is the median filter that returns the median intensity value in an $m \times m \times m$ neighborhood. However, more typical smoothing operators are often rep-

resented as 3-D convolutions with spatial lowpass masks which filter out high-frequency image components. The mask represents a weighted average of the intensity values in a $m \times m \times m$ neighborhood of each point in the volume. In addition to the mask size m, several other parameters may also be defined to adjust the level of smoothing and blurring by manipulating the weights for the averaging. One example is the 3-D Gaussian smoothing defined by a Gaussian mask with parameters $\sigma_1, \sigma_2, \sigma_3 \in (0, +\infty)$:

$$T[i,j,k] = e^{-\left(\frac{i^2}{2\sigma_1^2}+\frac{j^2}{2\sigma_2^2}+\frac{k^2}{2\sigma_3^2}\right)}$$

Sharpening operations aim to enhance geometric features by emphasizing the high-frequency components of the images. This can be achieved by applying a highpass filter, such as the Laplacian-type filter

$$f(x,y,z) = g(x,y,z) - \nabla^2 g(x,y,z)$$

to the image volume. Another useful operation is the unsharp masking that blends the low-frequency component and high-frequency component of an image volume, where the weights of the linear combination are adjustable parameters and represent the level of sharpening it generates. Again, most of these highpass filters can be represented by convolution masks as neighborhood functions.

An example is shown in Figure 3 where the structure in an actin filament volume is visualized through enhancement operations.

Boundary detection for surface rendering: Boundary detection operation finds the surface boundary voxels to derive the appropriate transfer function or thresholds for surface rendering. Most 2-D edge detection algorithms can be extended for 3-D boundary detection. Many of these algorithms employ some convolution masks to compute the discrete approximations of some differential operators to measure the rates of changes of the intensity field (gradients), and then classify surface boundary voxels based on a magnitude thresholding of the gradient values. More sophisticated edge detection algorithms have also been developed in Computer Vision (Perona and Malik 1990).

Iso-surface-based approach: Iso-surface rendering requires predefined iso-values to identify the iso-surfaces. Unfortunately, these iso-values are often not known in advance. Using the image-based approach, we can apply an edge detection operator to derive these iso-values automatically for iso-surface rendering. Note that extraction of the iso-surfaces (Lorensen and Cline 1987) is not necessarily needed. For instance, we can define a transfer function through intensity modification that renders only a layer of the surface voxels, defined by some narrow intensity intervals surrounding these iso-values.

To derive the iso-values, a histogram of all the boundary voxels from the boundary detection operator is first generated. The intensity values at which the histogram reaches local maxima can then be used as the surface iso-values. A smoothing operation (e.g., Gaussian smoothing) may need to be applied to the histogram first to remove noises. It should be mentioned that, with this approach, both the boundary detection and histogram analysis are precomputations of the actual rendering process. It is, therefore, possible to apply higher order edge detection operations in this process. By setting different scales of parameters in the boundary detection process, a set of multiscale iso-values can also be precomputed, and then used to define a set of multiscale transfer functions (as simple intensity tables) for different levels of surface rendering in data exploration.

Dynamic boundary detection-based approach: A second approach in using boundary detection for surface rendering is the direct application of a boundary detection operation to each sample point when it is accessed by the rendering algorithm. This allows the rendering algorithm to

(a) (b) (c)

FIG. 3 A fluorescently labeled actin filaments volume: (a) Volume rendering by a linear ramp transfer function, (b) Laplacian masking followed by unsharp masking with γ=3, (c) Laplacian masking followed by unsharp masking with γ=10.

determine dynamically whether or not a sample point belongs to a surface boundary for appropriate rendering actions. With this approach, only simple boundary detection methods (e.g., convolution mask-based detectors) ought to be applied for speed reasons. This approach is particularly useful for surfaces that cannot be simply defined as iso-surfaces. One example is the photobleaching effect in fluorescence microscopy, where the entire depth of the sample is illuminated with light that both excites and destroys fluorophores through photo-oxidation. When one attempts to collect serial optical sections of a sample volume, the images are characterized by an increase in the amount of photobleaching of each sequential plane, such that the same material may have different intensity values in different slices. In these cases, the boundaries may need to be identified using more sophisticated edge detection methods, and the data gradient is a more effective measure of surfaces than the iso-value. As a very simple example, the magnitudes of gradients may be proportionally mapped to the opacity values in the opacity transfer function to emphasize high-gradient regions for surface rendering effect. A gradient thresholding may also be used to render only the high gradient voxels. In general, this approach requires the intensity mappings for transfer function definition to be represented as neighborhood functions and is therefore more expensive than the iso-surface-based approach.

An example is shown in Figure 4 where the surfaces in a Golgi Complex volume are visualized through both iso-surface rendering and dynamic boundary detection. The iso-value is determined using the histogram curve of the boundary points.

Results

Using the techniques described in this paper, we have developed an integrated system, called IVIE, for the interactive visualization and imaging of volume data sets. Although IVIE is designed for general volume data, it is

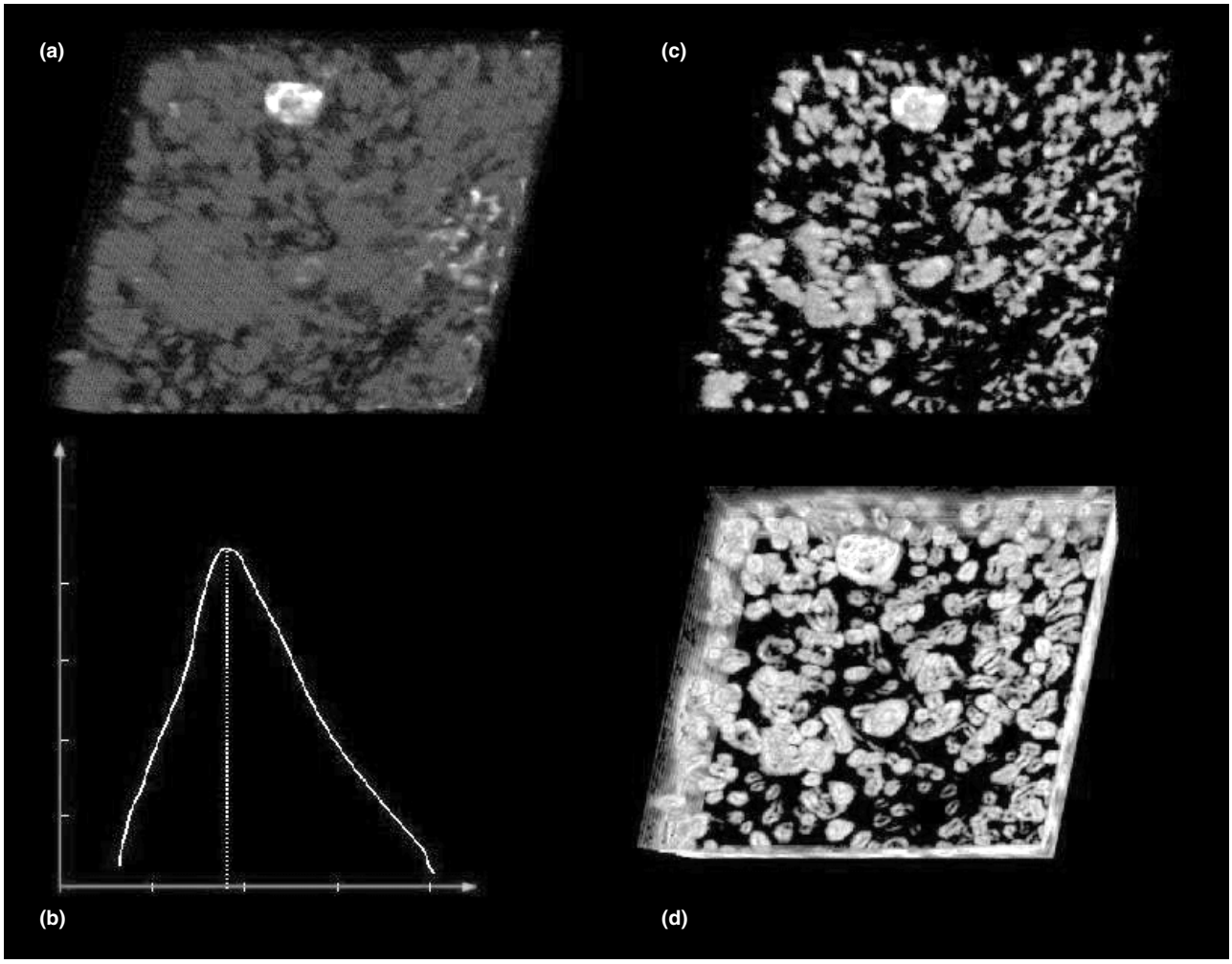

FIG. 4 Surface rendering of a Golgi complex: (a) Volume rendering by a linear ramp; (b) the histogram of boundary points; (c) iso-surface rendering with an iso-value 87, obtained from (b); (d) surface rendering by dynamic boundary detection.

particularly suitable for noisy and complex data sets, such as 3-D microscopy data volumes, that require sophisticated transfer functions and interactive data exploration. The system is written in C++ and OpenGL 1.1, and is currently implemented on SGI OCTANE and O2 workstations. Since only 2-D texture mapping hardware is used, it can be easily ported to PC platforms. Interactive volume rendering is achieved using the 2-D texture mapping based shear-warp algorithm described in the section titled "Interactive Volume Rendering by 2-D Texture Mapping." The algorithm requires a 128 MB main memory to run 256^3 sized volumes. Unlike the original shear-warp algorithm, this system shows no speed difference between parallel and perspective projections. On an SGI OCTANE workstation with 128 MB main memory and 4 MB texture memory, we are able to volume render a $256 \times 256 \times 64$ data set at a 7

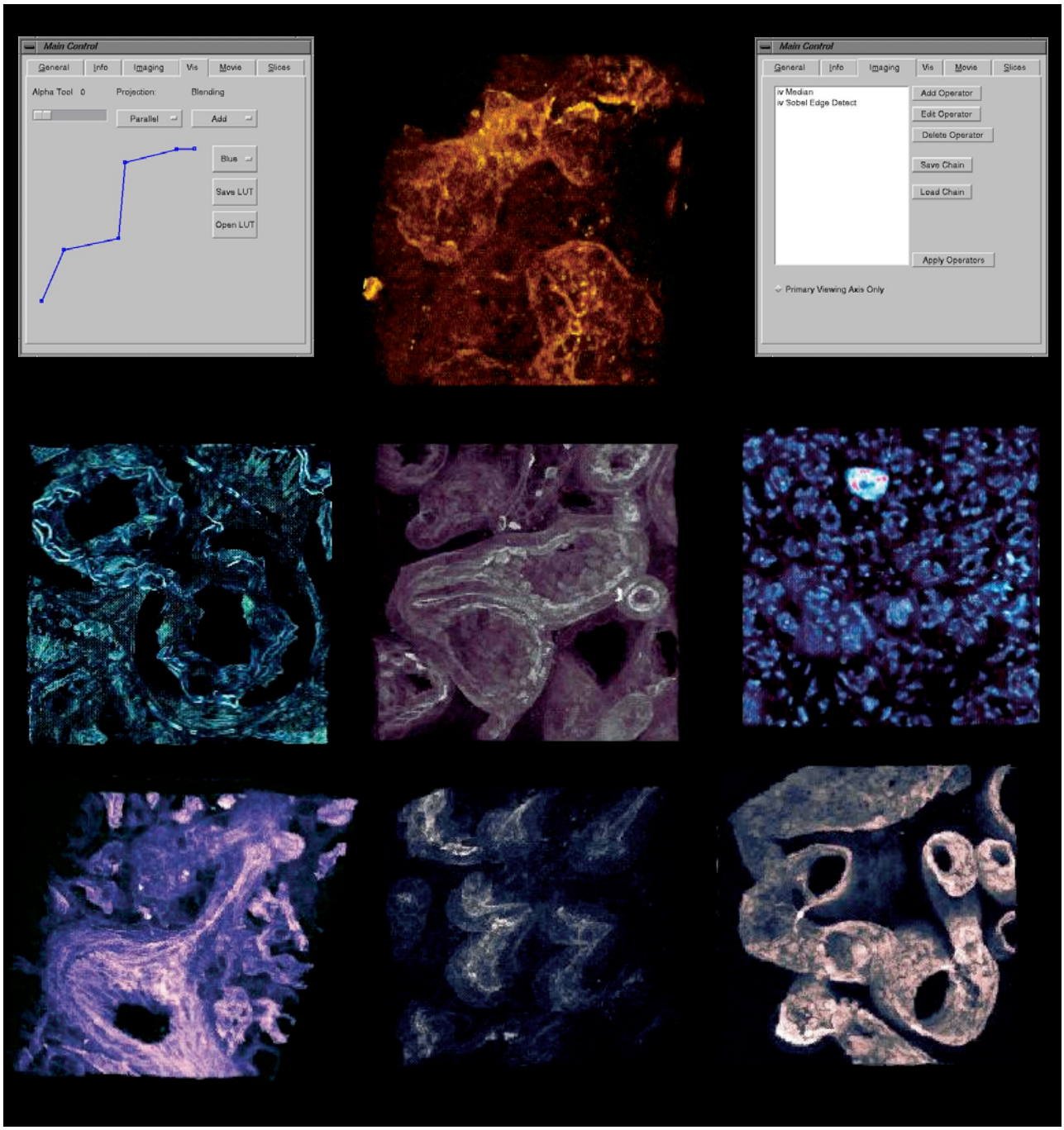

FIG. 5 Color images: Interactive volume visualization system (IVIE) rendering results of several confocal microscopy data sets, and two sample control screens.

frames/s rate. The frame rate is reduced to 5 frames/s on an SGI O2 workstation with 128 MB main memory and 1 MB texture memory. Such frame rates allow users to rotate and zoom in/out interactively with full volume rendering resolution. For an even better frame rate, IVIE also provides an adaptive rendering option that automatically adjusts the number of slices (2-D textures) to be used for rendering when the object is in motion. Since loss of resolution during motion is normally less noticeable, it is a very effective feature for smooth animation and motion.

Two image processing libraries are used in IVIE. The first is a software toolkit we wrote using C++ for both 2-D and 3-D image processing operations. The second is the *ImageVision* library by SGI. Since *ImageVision* takes advantages of some hardware features through OpenGL, it is faster than our software implementations. On the other hand, it is only available on SGI workstations, and would need to be replaced by either a software library or some hardware image processing chips if the system is ported to a PC platform. For $256 \times 256 \times 64$ volumes, the transfer function computation without *ImageVision* takes from 6 to 170 s for each filter. It is about twice as fast if *ImageVision* were used. Although the image processing computation using *ImageVision* is very fast, the majority of the time for the transfer function process is spent on building and loading the 2-D texture objects rather than on the image processing itself.

The front end of IVIE is a graphical user interface with a rendering window and a control panel that includes a number of control screens for controlling different system functions, such as volume rendering, 2-D and 3-D image processing, color and opacity map manipulation, transparency adjustment, movie making, and slice-based operations. Some sample control screens and rendering images generated by IVIE are shown in Figure 5.

Conclusions

An interactive volume visualization system, IVIE, for 3-D microscopy images is described. Two new techniques are employed in this system: interactive volume rendering by 2-D texture mapping and transfer function design using image processing operations. They form the basis of an interactive volume data exploration environment that is particularly suitable for 3-D microscopy data sets. The system is intentionally designed to avoid using advanced graphics hardware features such as 3-D texture mapping. Thus, it can be easily implemented on PC platforms and other low-end workstations. Future work will be focused on the performance enhancement of the image processing computation and the PC implementation of the IVIE system. Currently, the speed of the image processing operations in IVIE is not yet interactive. We would like to investigate new image processing solutions (possibly hardware enhanced) to achieve truly interactive transfer function design. Since the graphics accelerators on PCs often have very large 2-D texture memory, we expect that the performance of our volume rendering algorithm on PCs will be even better.

References

Cabral B, Cam N, Foran J: Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *Proc. 1994 Symposium on Volume Visualization,* 91–98, October (1994)

Cameron GG, Undrill PE: Rendering volumetric medical images data on a simd-architecture computer. In *Proc. 3rd Eurographics Workshop on Rendering*, 135–145, May (1992)

Dunn K, Mayor S, Meyer J, Maxfield F: Applications of ratio fluorescence microscopy in the study of cell physiology. *FASEB J* 8, 573–582 (1994)

Fang S, Biddlecome T, Tuceryan M: Image-based transfer function design for data exploration in volume visualization. In *Proc. IEEE Visualization '98*, 319–326, October (1998)

He T, Hong L, Kaufman A, Pfister H: Generation of transfer functions with stochastic search techniques. In *IEEE Visualization 96*, 227–234, October (1996)

Kindlmann G, Durkin J: Semi-automatic generation of transfer functions for direct volume rendering. In *IEEE/ACM 1998 Symposium on Volume Visualization* (1998)

Lacroute P, Levoy M: Fast volume rendering using a shear-warp factorization of the viewing transformation. *SIGGRAPH '94,* 451–458 (1994)

Levoy M: Display of surfaces from volume data. *IEEE Computer Graphics and Application,* 8(3), 29–37, May (1988)

Levoy M: Efficient ray tracing of volume data. *ACM Trans on Graphics*, 9(3), 245–261, July (1990)

Lorensen WE, Cline HE: Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics, SIGGRAPH '87,* 21(4), 163–169, July 1987

Marks J, Andalman B, Beardsley PA, Freeman W, Gibson S, Hodgins J, Kang T, Mirtich B, Pfister H, Ruml W, Ryall K, Seims J, Shieber S: Design galleries: A general approach to setting parameters for computer graphics and animation. In *SIGGRAPH '97,* 389–400 (1997)

Parker S, Shirley P, Livnat Y, Hansen C, Sloan P: Interactive ray tracing for isosurface rendering. In *Proc. IEEE Visualization '98,* 233–238 (1998)

Perona P, Malik J: Scale-space and edge detection using anisotropic diffusion. *IEEE Trans Pattern Analysis and Machine Intelligence,* 12(7) 629–639 (1990)

Rosenfeld A, Kak A: *Digital Picture Processing*. Academic Press (1982)

SGI Technical Publications. *Open GL Volumizer Programmer's Guide.* SGI (1998)

Shaw PJ: Comparison of wide-field/deconvolution and confocal microscopy for 3D imaging. In *Handbook of Biological Confocal Microscopy,* 2nd Edition, (1995) 373–387

Upson C, Keeler M: V-buffer: Visible volume rendering. *Computer Graphics, SIGGRAPH '88,* 22(4), 59–64, August (1988)

Westover L: Footprint evaluation for volume rendering. *Computer Graphics, SIGGRAPH '90,* 24(4) 367–376, August (1990)