

Image-Based Transfer Function Design for Data Exploration in Volume Visualization

Shiaofen Fang

Tom Biddlecome

Mihran Tuceryan

Department of Computer and Information Science
Indiana University Purdue University Indianapolis

ABSTRACT

Transfer function design is an integrated component in volume visualization and data exploration. The common trial-and-error approach for transfer function searching is a very difficult and time-consuming process. A goal-oriented and parameterized transfer function model is, therefore, crucial in guiding the transfer function searching process for better and more meaningful visualization results. This paper presents an image-based transfer function model that integrates 3D image processing tools into the volume visualization pipeline to facilitate the search for an image-based transfer function in volume data visualization and exploration. The model defines a transfer function as a sequence of 3D image processing procedures, and allows the users to adjust a set of qualitative and descriptive parameters to achieve their subjective visualization goals. 3D image enhancement and boundary detection tools, and their integration methods with volume visualization algorithms are described in this paper. The application of this approach for 3D microscopy data exploration and analysis is also discussed.

Keywords – volume visualization, 3D image processing, transfer function, volume rendering, data exploration.

1 INTRODUCTION

Volume visualization is a field in scientific visualization that is concerned with the abstraction, interpretation, rendering and manipulation of large volume datasets. It has been widely used in many scientific, engineering and biomedical applications. Two typical volume visualization techniques are volume rendering and surface rendering. Volume rendering algorithms[9, 2, 16] can directly display the volume information contained in the dataset through semi-transparent images, and thus allow the users to explore the internal structures of the image volume. One key component in this process is the transfer function that maps the volume's intensity values to color and opacity values for display. In surface rendering[10], the transfer function defines the thresholds with which iso-surfaces are extracted and rendered as surface objects. In both cases, the transfer function is used as both a filter that selects a subset of the volume information to be presented, and an information interpreter that determines how the selected information is to be presented.

The need for transfer function design and modeling comes from the dynamic and often subjective visualization goals and requirements in different applications. Users often need to interactively search and manipulate the transfer function in the visualization process to view different sets of sub-structures, surfaces and frequencies in desired ways. For applications such as medical CT image viewing, this may not be a difficult problem, since the CT intensities of most medical objects (e.g. bones and tissues) are generally

known. Even so, many advanced applications still need to visualize the finer anatomic structures that cannot be easily obtained with trivial transfer functions. A more challenging situation is when the volume dataset contains objects that are highly complex with various levels of signal-to-noise ratios. Moreover, the structures embedded within the dataset can be partially or entirely unknown. One example is the visualization of microscopy image volumes for the study of cell biology, where many of the cellular structures, their optical/material properties, and the associated image signals are highly complex, noisy and often unknown[1]. In such cases, an efficient and goal-oriented transfer function model is crucial in exploring the unfamiliar data environment for scientific discovery, data analysis, or medical diagnosis.

The most common approach in transfer function design is by *trial-and-error*. It allows the user to arbitrarily and repeatedly manipulate the coefficients of some mathematical representation of the transfer function to adjust the visualization outcome. Common forms of such mathematical representations are piecewise linear functions, polynomials and splines. In practice, however, such a *trial-and-error* approach is very difficult and time-consuming without prior knowledge about the optical and material properties of the underlying image volumes. This is mainly because the coefficients of the transfer function representation has little or no pre-defined qualitative meanings or visual relationships to the rendered images. As a result, there is very little guidance that can be followed in the transfer function searching process, and the visualization outcome, thus, largely depends on the user's experience and "luck". Since such a *trial-and-error* process needs to be done all over again for each new image volume or visualization objective, it puts an enormous burden to the application users. Furthermore, such "arbitrary" transfer function manipulation without proper constraints can easily lead to confusing, misleading and dubious visualization results.

Previous effort in improving the transfer function searching is rather limited. One interesting work is by He *et al*[6]. It uses a stochastic search technique to generate many image samples based on an initial population of pre-defined transfer functions. Users are then allowed to select the proper sample images, based on visual examinations, to assist the filtering and evolution of the transfer function population. A similar approach has also been taken in the *Design Galleries* work by Marks *et al*[11]. Although this approach provides some level of heuristics for transfer function searching, it is still based on some mathematical representations with parameters that are not directly and intuitively related to user's visualization objectives. Moreover, simply selecting sample images appears to be a very inefficient and unnatural way of expressing the user's visualization goals. The limitations in the number of sample images, their viewing perspectives and the time that is needed to generate the sample images also severely constrain the applicability of this approach. Another related topic is volume shading[8, 12]. Often,

the gradient of each point in the volume is used as the normal vector of an imaginary local surface which can be lighted with some surface shading model. It assigns colors that depend on the local image properties (gradients), and can effectively reveal structural details that are related to the gradient field of the volume. As described in this paper, image dependent transfer functions can also be used to achieve many other visualization goals such as information filtering and surface rendering.

The objective of this work is to develop an image-based, goal-oriented, and parameterized transfer function model for volume visualization and data exploration. This is done by integrating 3D image processing tools into a volume visualization process, and defining the transfer function to be a sequence of 3D image processing procedures as part of the volume visualization pipeline. Two major types of 3D image processing tools[14], image enhancement and boundary detection, are applied. Users will be able to specify and manipulate a small number of parameters to define their visualization goals, and to view desired features in various levels of details. Since these parameters represent the parameter settings of the image processing procedures with clear meanings and objectives, it provides an efficient, image-guided and goal-oriented interaction for data exploration. Although image processing techniques have previously been applied to the segmentation of 3D CT and MRI image volumes[7, 15, 4, 5], and image gradient information has also been widely used for volume and surface renderings[8], there has not been a general and systematic solution for the integration of 3D image processing in transfer function design.

In the following, we will first present, in Section 2, the image-based transfer function model, and the schemes that integrate this model into the volume rendering process. Section 3 describes the various image processing operations as the building blocks of the transfer function model, including point enhancement, spatial enhancement and boundary detection. Some implementation issues, and the application of this approach in a microscopy data exploration project are discussed in Section 4. Section 5 concludes the paper with some final remarks and future work.

2 AN IMAGE-BASED TRANSFER FUNCTION MODEL

We define a *volume* to be a 3D array of voxels in an intensity (scalar) field, and a *color volume* (or *RGBA volume*) to be a 3D array of voxels in an RGBA (color and opacity) field. A *color volume* can be directly displayed using a volume rendering algorithm such as raycasting[9] or 3D texture mapping[3], without the need for a transfer function. Therefore, we define a transfer function to be a two step process. The first step consists of a sequence of intensity mappings in a volume’s intensity field, and performs tasks such as information filtering, noise reduction, surface extraction, etc. The second step is essentially the “*coloring*” process, which generates colors and opacity values directly from intensity values using either an intensity-to-RGBA color look-up table or a shading procedure. In the color look-up table, a linear ramp should be used for the opacity component and the appropriate (depending on the desired colors) color components, since the necessary “intensity processing” is done in the first step. Shading can be done by computing the gradients of the resulting intensity field of the first step, and using the gradients as local “surface normals” in computing the lighting effects in the rendering process. Since the “*coloring*” step is a fairly straightforward process, we will only consider the transfer function design problem in the volume’s intensity field.

Based on the above definition, a *volume* is essentially a 3D gray level image, and its transfer function problem can be considered as a 3D gray level image processing problem. Naturally, various image processing tools can be applied, each having a specific image pro-

cessing objective with user defined parameters. In addition to providing a goal-oriented model for transfer function design, the combination of 3D image processing and volume visualization also offers some other unique advantages. First, when integrated into a visualization pipeline, the 3D image processing operations only need to be applied to the visible regions of the volume. This may lead to significant savings of the image processing cost. Secondly, many traditionally difficult problems in image processing and computer vision, such as the topological consistency and connectivity problems in boundary detection, may become trivial in an interactive visualization environment, where certain “intelligent” decisions are left to the viewers. Thirdly, as described in [15], interactive visualization may also provide useful human intervention to assist some difficult image processing tasks such as 3D segmentation.

An image-based transfer function, $F : \mathcal{I} \rightarrow \mathcal{I}$, can be defined as a sequence of intensity mappings:

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

where \mathcal{I} is the volume’s intensity domain, $f_i : \mathcal{I} \rightarrow \mathcal{I}$ are intensity mappings corresponding to the sequence of image processing procedures applied. In our transfer function model, each f_i is 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. Linear interpolation will need to be applied for each input intensity value that is not one of the table entries.
2. *Neighborhood function*. It is a function computed from the intensity values in a $m \times m \times m$ neighborhood of a given voxel, where the neighborhood size, m , can be an adjustable parameter of the transfer function. A *median filter*[14], for instance, can be considered as a *neighborhood function*. A more typical example is the 3D spatial convolution, as a linear 3D filter, of a volume V with a $m \times m \times m$ 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 and global image processing operations, such as *dilation/erosion* and *anisotropic diffusion*[13], cannot be directly represented as a *neighborhood function*. But these operations are normally applied in some pre-computation processes for the definitions of other simpler intensity mappings, as shown in Section 3.3.

Each of the intensity mappings, f_i , represents an image processing procedure with user defined parameters, which computes either an *intensity table* or a *neighborhood function*. The set of all parameters of a sequence of image processing procedures defines one transfer function in the transfer function space. Such goal-oriented parameterization of the transfer function model makes the searching and navigation of the transfer function space much more user friendly and intuitive than previous methods.

There are three different approaches, as shown in Figure 1, of applying this transfer function model in volume visualization. In the first approach, the transfer function is defined as a sequence of 3D image processing procedures through a sequence of intermediate volumes, and the final resulting volume is passed to the “*coloring and rendering*” step. As shown in Figure 1(a), this approach requires the reconstruction of an intermediate volume for each image processing procedure. Although this approach is easy to implement, its obvious drawback is the high cost in memory use and computational time for volume reconstructions. Since data exploration requires experimenting with many different sets of parameters for

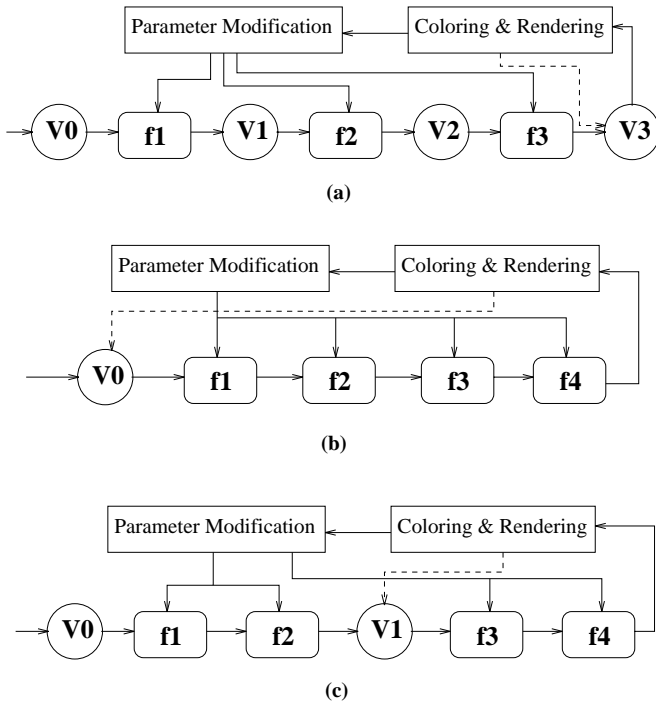


Figure 1: (a) The volume reconstruction based approach; (b) The fully integrated approach; (c) The hybrid approach

various visualization goals, constructing new intermediate volumes for every change of parameters may not be practical.

The second approach integrates all the image processing procedures into one volume rendering algorithm, so that every access to an intensity value in the volume rendering algorithm will directly go through the computations of all the 3D image processing procedures (Figure 1(b)). In addition to avoiding the large memory requirement and the associated memory operation costs, this approach applies the image processing procedures only when a sample point in the volume is actually used by the visualization algorithm for rendering. Since the number of sample points used for rendering is normally much smaller (less than 10% in most cases) than the total number of voxels in the volume, such integrated approach is, in general, a more efficient solution for interactive data exploration applications where the change of transfer function parameters is very frequent, and the interactive renderings are often lower resolution and “region of interest” based.

The integration of the *intensity tables* into a visualization pipeline is very straightforward. The *intensity table* can be directly used as the opacity and/or color transfer functions in any volume rendering algorithm (e.g. raycasting). In volume rendering using 3D texture mapping, it can also be conveniently used as a *color table*, such as the one defined in OpenGL, that maps (often by hardware) the texture mapping results to color and opacity values. Very little computational overhead is involved in this process.

The integration of *neighborhood functions* are, however, more costly. A straightforward approach is to make recursive procedural calls to the *neighborhood functions* to dynamically compute the image processing results for individual sample points when they are accessed by the rendering algorithm. One problem with this approach, however, is the potentially repeated computation with multiple *neighborhood functions*. Since each voxel can fall into the neighborhoods of several other voxels, and may therefore be accessed (and computed) multiple times when more than one neighborhood functions exist in a transfer function. When the number of *neigh-*

borhood functions in a sequence is large, such overhead can be significant. Fortunately, this has not been a major problem in most cases for two reasons. First, the number of image processing procedures applied at one time is normally small in most practical applications. Secondly, since the number of sample points that need to be computed by the transfer function for each rendering is not very large (normally less than 10% of the total number of voxels in a volume), it is possible to apply some buffering mechanism to store the result of each computed sample point for possible later accesses (e.g. a buffer can be used for each *neighborhood function*). This way, the overhead can be partially or entirely eliminated depending on the buffering capacity.

The third approach uses a hybrid strategy, and allows intensity mappings to be concatenated into groups to generate intermediate volumes (Figure 1(c)). The grouping of the intensity mappings can either be defined by users or done automatically based on their computational complexities. This is mainly useful for transfer functions with long sequences of *neighborhood functions*. One advantage of the hybrid approach is that the parameters of the image processing procedures are naturally defined in groups. For example, an intermediate volume can be generated when a satisfactory set of parameters for a group have been found, and the subsequent image processing operations may then be based on the intermediate volume instead of the original volume.

3 IMAGE VOLUME OPERATIONS IN TRANSFER FUNCTIONS

In this section, we will describe various 3D image processing operations and their applications in transfer functions for volume visualization. We are mainly interested in two major types of operations: image enhancement (including point enhancement operations and spatial enhancement operations) and boundary detection[14]. The goal of image enhancement is to improve the quality of the 3D image volume for better visual appearance, based on certain visualization goals. Boundary detection, on the other hand, automatically finds the voxels that belong to some surface boundary defined by the parameters of the edge detection procedures, and is mostly used for surface rendering purpose.

3.1 Point Enhancement Operations

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 on which it is applied, the corresponding intensity mapping can be represented as an *intensity table*, obtained from a pre-computation of an image processing operation under given parameters. The two most common point enhancement operations are *Intensity modification* and *Histogram modification*.

Intensity Modification: In *intensity modification*, the intensity curve of the input volume can be altered by modifying one or more segments of the intensity curve at corresponding intensity intervals to increase or decrease the exposure of the given regions. In Figure 2(a), for example, interval $[t_1, t_2]$ is stretched to expose more details within this intensity range. Although t_1, t_2 and r can all be used as parameters of the transfer function, they more likely come from the output of some other image processing procedures, such as boundary detection for surface rendering, as shown in Figure 2(b) and explained in Section 3.3.1.

Histogram Modification: *Histogram modification* changes the histogram curve of a volume to generate a re-distribution of the intensity values. One particularly important and useful operation is

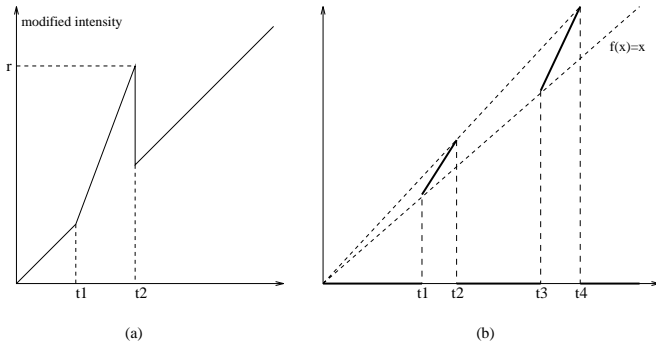


Figure 2: (a) A simple intensity modification; (b) Intensity modification for surface rendering

the *histogram equalization*, which flattens the histogram to increase the contrast in the areas with large number of voxels concentrated around certain values. *Histogram equalization* does not require parameters.

3.2 Spatial Enhancement Operations

A spatial enhancement operation derives the new intensity value of a given point from its neighborhood points, i.e. the result is region dependent. Therefore, spatial enhancement operations, in general, can only be represented as *neighborhood functions*. As in the 2D case[14], spatial operations can be classified into *smoothing* and *sharpening* operations.

Smoothing Operations: We use *smoothing* operations primarily to remove image noise. We sometimes also want to remove very small feature details in order to better present the larger features. These are often achieved by applying a spatial *lowpass* 3D mask to smooth out high frequency components from the image. The mask represents a weighted average of the intensity values in a $m \times m \times m$ neighborhood of each point in the volume. For smoothing purpose, all coefficients in the masks are positive numbers. 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. A popular smoothing operation is the *Gaussian* smoothing defined by a *Gaussian* mask with parameter $\sigma \in (0, +\infty)$:

$$T[i, j, k] = e^{-\frac{i^2 + j^2 + k^2}{2\sigma^2}}$$

The parameter σ may also be different in X , Y and Z directions:

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

Another useful smoothing operation is the *Median filter* that returns the median intensity value in a $m \times m \times m$ neighborhood. An advantage of the *Median filter* is that it can avoid blurring edge and surface boundaries, while still being able to achieve effective noise reduction.

Sharpening Operations: *Sharpening* operations aim to enhance of geometric features by emphasizing the high frequency components of the images[14]. This can be achieved by applying a *highpass* convolution mask to the image volume. An example is the *Laplacian*-type filter:

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

which can also be represented as a $m \times m \times m$ mask. Another useful operation is the *unsharp masking* that blends the low-frequency

component V_1 and high-frequency component $(V - V_1)$ of an image volume V :

$$V' = \gamma \cdot (V - V_1) + V_1 = \gamma \cdot V + (1 - \gamma) \cdot V_1$$

where coefficient $\gamma \in (0, +\infty)$ represents the amount of high-frequency component used in the enhanced image volume V' . $\gamma \in (0, 1)$ results in a smoothing of V , and $\gamma > 1$ emphasizes the high-frequency component, and therefore sharpens features and details. V_1 can be obtained from V by applying a smoothing operation to V . For instance, if a simple $3 \times 3 \times 3$ averaging mask is applied to compute V_1 , the overall convolution mask for this *unsharp* operation will be:

$$T[i, j, k] = \begin{cases} (26\gamma + 1)/27 & \text{if } i = j = k = 0 \\ (1 - \gamma)/27 & \text{otherwise} \end{cases}$$

Since *sharpening* operations tend to enhance high frequency intensity signals, including noise which usually exhibit strong high frequency characteristics, a *smoothing* operation may need to be applied first to remove or reduce the noise before the *sharpening* operation. Some examples are shown in Figure 4 and Figure 5.

3.3 Boundary Detection Operations For Surface Rendering

Boundary detection operation finds the surface boundary voxels to derive the appropriate transfer function or thresholds for surface rendering. Most 2D edge detection algorithms[14] can be extended for 3D boundary detection. Many of these algorithms, such as the *Sobel* detector[14], 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. One example is the class of *anisotropic diffusion* algorithms that successively blur the image using a diffusion process with a spatially varying conductance parameter that is correlated to the image's gradient values[13]. A general form of this operation is:

$$I_t = \text{div}(c(x, y, z, t) \nabla I) = c(x, y, z, t) \nabla^2 I + \nabla c \cdot \nabla I$$

where the conductances $c(x, y, z, t)$ are usually taken to be high in uniform regions and low at high gradient points in the image. This results in the image being smoothed in uniform regions but the discontinuities (edges) be kept at the high gradient points. If the conductance function $c(x, y, z, t)$ is constant, then the result is essentially a *Gaussian* blurring process.

3.3.1 Iso-surface Based Approach

Conventional surface rendering algorithms[10] first extract the surface boundaries as polygonal objects by intensity thresholding, and then display the surface boundaries using standard surface graphics techniques. In many cases, however, the intensity thresholds defining the iso-surfaces are not known in advance. Finding these thresholds by trial-and-error is difficult and time-consuming, particularly when the volume contains many different types of object boundaries with different iso-values. Using the image-based approach, we can first apply an edge detection operation to the image volume, and then automatically derive, based on the edge detection results, the surface boundary iso-values for iso-surface extraction. These iso-values may also be used to defined a transfer function through *intensity modification* with a standard volume rendering algorithm that renders only a layer of surface voxels, defined by intensity intervals

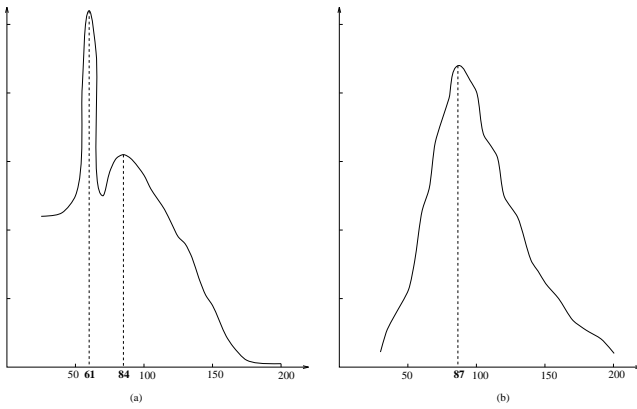


Figure 3: Histograms of surface voxels after boundary detection from: (a) a CT volume of a human head; (b) a microscopy volume of a Golgi Complex.

surrounding these iso-values. Figure 2(b) shows an example where the boundary intensity range, $[t1, t2]$ and $[t3, t4]$, are stretched to highlight the surfaces, and the intensity values of the rest of the intensity field is reduced to zero.

To derive the surface intensity thresholds after boundary detection, we first use a simple thresholding on the boundary detection result to extract all the boundary voxels, and then generate a histogram based on the intensity values of all boundary voxels in the original volume. For volumes with well defined surfaces, this histogram should exhibit clear *peaks and valleys*. The intensity values at which the histogram reaches local maxima (peaks) can then be used as the surface thresholds. A smoothing operation (e.g., *Gaussian* smoothing) normally needs to be applied to the histogram first to remove the noise. Two examples are shown in Figure 3, generated from a CT volume of a human head and a microscopy image volume of a Golgi Complex (Figure 6). In Figure 3(a), the histogram has two local maxima at intensity values 61 and 84 that represent the iso-values of the skin and skull surfaces, respectively. Figure 3(b) shows one local maximum, at intensity value 87, for the Golgi surface boundaries.

It should be mentioned that, with this approach, both the boundary detection and histogram analysis are pre-computations of the actual rendering process. By setting different scale parameters in the boundary detection process, a set of multiscale iso-values can be pre-computed, and then used to define a set of multiscale transfer functions (as simple *intensity tables*) for different levels of surface rendering in data exploration.

3.3.2 Dynamic Boundary Detection Based Approach

A second approach in using boundary detection for surface rendering is to directly apply a boundary detection operation to each sample point when it is accessed by the rendering algorithm. This allows the rendering algorithm to dynamically determine whether a sample point belongs to a surface boundary or not for appropriate rendering actions. With this approach, only simple boundary detection methods (e.g. convolution mask based detectors) ought to be applied for speed reason. This approach is particularly useful for surfaces that cannot be simply defined as iso-surfaces, i.e. the boundary intensity values are not constants. Usually, such non-constant-intensity boundaries are caused by the data collection process. In 3D microscopy, for instance, photobleaching may cause the same material to have different intensity values in separate slices with different depths into the volume. In this case, working directly with gradients is more effective than working with iso-surfaces. For

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. An example is given in Figure 6(b) which shows better defined surfaces than iso-surface based rendering (Figure 6(c)). In general, this approach requires the intensity mappings for transfer function definition be represented as *neighborhood functions*, and is therefore more expensive than the iso-surface based approach.

4 IMPLEMENTATION AND APPLICATIONS

The approach described in this paper is being implemented in a microscopy visualization and data exploration system – a joint project with the Department of Medicine of the Indiana University Medical School. Microscopy volumes offer some unique challenges to volume 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, signal-to-noise ratio decreases with the collection of each focal plane of an image volume. The resulting low contrast and small intensity gradients make these image volumes sensitive to small changes in rendering parameters. Consequently, ordinary volume visualization algorithms frequently fail to capture the delicate structures present in many cellular objects. Secondly, 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 multi-parameter images, in which several different proteins will be imaged simultaneously, each in a specific color of fluorescence. A third problem is that the structure of the objects to be examined are often partially or entirely unknown. Without prior knowledge of the structures in an image, it is difficult to determine an appropriate transfer function to generate a useful and informative rendering.

As a result, we found that, without an interactive and image-based environment for transfer function design, it is often impossible to generate useful images from a complex, noisy and unknown microscopy image volume. We also found that, for CT images of human anatomy, the enhancement operations do not have as significant improvements in visualization quality as they do with microscopy images. This is perhaps because the CT images we tested are already very clean and the transfer functions are fairly obvious for the common visualization goals.

In integrating image-based transfer functions into volume visualization algorithms, the first two approaches given in Section 2 have been implemented and tested using a standard raycasting volume rendering algorithm with shading. Volume rendering using hardware assisted 3D texture mapping has also been implemented in this context. But since the hardware rendering pipeline with 3D texture mapping cannot be easily modified by the algorithm, only the first approach was used with the texture mapping algorithm. The volume reconstruction based approach is, in general, fairly slow. For instance, reconstructing a 256^3 volume with a $5 \times 5 \times 5$ mask takes about 4 minutes on an SGI O2 (R5000) workstation. If the convolution is integrated into the rendering pipeline, only about 10 seconds need to be spent on computing the convolution. In the case of multiple *neighborhood functions* in a transfer function, if a separate buffer for each *neighborhood function* is used to avoid repeated computation, the overhead from the integration is very little, i.e. the integrated approach is always faster. But if no buffering is applied, the overhead is significant — usually when there are more than three *neighborhood functions*, the integrated approach quickly becomes slower than the image reconstruction based approach. It should be mentioned that the *intensity tables* do not have any noticeable im-

pect to the rendering speed in the integrated approach.

Several rendering examples of confocal microscopy volumes using the image-based transfer function model are given below.

Figure 4 shows the rendering of a microtubule dataset using image enhancement operations. Microtubules are the molecular tracks within cells. Intracellular organelles or compartments are spatially distributed inside the cell but this spatial distribution is dynamic. The organelles can move along these tracks in molecular motors. Figure 4(a) is rendered with a linear ramp transfer function. Figure 4(b) is rendered with a *median filter* applied first. In Figure 4(c), an additional *highpass filter*, the *Laplacian filter*, is applied after the median filter. The result shows the tubule structures that are not clearly visible from other renderings.

Figure 5 shows the effect of image enhancement in the volume rendering of fluorescently labeled Actin filaments volume. An *Laplacian* mask is applied followed by an *unsharp masking* operation with $\gamma = 3$ (Figure 5(b)) and $\gamma = 10$ (Figure 5(c)) to show the actin structures that are not at all visible from the rendering with a linear ramp (Figure 5(a)).

Figure 6 shows two surface rendering examples of a Golgi Complex. Figure 6(a) is rendered with a linear ramp transfer function. The surface rendering by the dynamic boundary detection based approach is shown in Figure 6(b). A gradient magnitude threshold 60 was used to classify the boundary voxels. Figure 6(c) is rendered by the iso-surface based approach with an intensity interval [77, 97] centered at the surface iso-value 87, which is derived from the boundary histogram (Figure 3(b)) using the same boundary detection procedure. For this dataset, the dynamic boundary detection based approach generates better surfaces.

We have also applied the boundary detection approach for surface rendering on a CT volume of a human head. Figure 6(d) is rendered by the dynamic boundary detection based approach. A gradient magnitude threshold 40 was used. The surface iso-values, 61 and 84, derived from the boundary histogram (Figure 3(a)), are then used to render Figure 6(e) and Figure 6(f), using intensity thresholding with intervals [53, 69] and [74, 94], respectively. In cases like this where more than one types of surfaces are present, the iso-surface based approach is more flexible since it can select individual surfaces to display, while the dynamic boundary detection based approach has to display all surfaces at once.

5 CONCLUSIONS

We have presented a transfer function model based on 3D image processing operations. In this approach, transfer functions are represented as a sequence of intensity mappings that can be either *intensity look-up tables* or *neighborhood functions*. The computation of these intensity mappings can be integrated into a volume rendering algorithm for better memory and time efficiency. Both 3D image enhancement operations and boundary detection operations are described and integrated into the transfer function design problem. The parameterization of this transfer function model allows goal-oriented interactions in transfer function searching and data exploration. This approach is particularly useful for volume datasets that are complex and noisy with unknown structures. It is our belief that static or pre-defined sequences of renderings cannot provide sufficient insight into a complicated image volume. It is the dynamic and interactive exploration process with guided user control that provides the most comprehensive perspective into the dataset.

In the future, we plan to embed a richer set of 3D image processing tools into volume visualization. One particularly important subject is 3D image segmentation applied for transfer function design. This will lead to object level transfer function definitions, and model-based volume representations, manipulations and analysis. Another important future work is the interactivity of this image-based

approach. Currently, the rendering speed is not interactive due to mainly the time spend on image processing operations. More efficient 3D image processing algorithms, and a spatially and temporally optimal buffering mechanism need to be developed for truly interactive data exploration. In terms of applications, we are currently developing an interactive environment for microscopy data exploration based on the approach described in this paper. A multiscale approach will be employed to provide a rich and well organized parameter hierarchy for transfer function searching and volume data exploration.

6 ACKNOWLEDGMENTS

We would like to thank Dr. Ken Dunn and Dr. Bob Bacallao from the Indiana University School of Medicine for providing the microscopy image volumes and the relevant biological background. We would also like to thank Yi Dai for implementing many of the image processing tools.

REFERENCES

- [1] Tom Biddlecome, Shiao-fen Fang, Ken Dunn, and Mihran Tuceryan. Image guided interactive volume visualization for confocal microscopy data exploration. In *Proc. 1998 SPIE International Symposium on Medical Imaging*, 1998.
- [2] J. Danskin and P. Hanrahan. Fast algorithms for volume ray tracing. In *Proc. 1992 Workshop on Volume Visualization*, pages 91–98, October 1992.
- [3] Shiao-fen Fang, Rajagopalan Srinivasan, Su Huang, and Raghu Raghavan. Deformable volume rendering by 3D texture mapping and octree encoding. In *Proc. of IEEE Visualization '96, San Francisco*, pages 73–80, October 1996.
- [4] G. Gerig, O. Kübler, R. Kikinis, and F.A. Jolesz. Nonlinear anisotropic filtering of MRI data. *IEEE Transactions on Medical Imaging*, 11(2):221–232, June 1992.
- [5] G. Gerig, J. Martin, R. Kikinis, O. Kübler, M. Shenton, and F.A. Jolesz. Unsupervised tissue type segmentation of 3D dual-echo MR head data. *Image and Vision Computing*, 10(6):349–360, July 1992. IPMI 1991 special issue.
- [6] T. He, L. Hong, A. Kaufman, and H. Pfister. Generation of transfer functions with stochastic search techniques. In *IEEE Visualization 96*, pages 227–234, Oct 1996.
- [7] K. H. Höhne and W. Hanson. Interactive 3D segmentation of MRI and CT volumes using morphological operations. *Journal of Computer Assisted Tomography*, 16(2):285–294, 1992.
- [8] Marc Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Application*, 8(3):29–37, May 1988.
- [9] Marc Levoy. Efficient ray tracing of volume data. *ACM Transactions on Graphics*, 9(3):245–261, July 1990.
- [10] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics, SIGGRAPH '87*, 21(4):163–169, July 1987.
- [11] J. Marks, and et al. Design galleries: A general approach to setting parameters for computer graphics and animation. In *Computer Graphics (SIGGRAPH '97 Proceedings)*, pages 389–400, 1997.
- [12] N. Max. Optical models for direct volume rendering. *IEEE trans. on Visualization and Computer Graphics*, 1(2):99–108, June 1995.

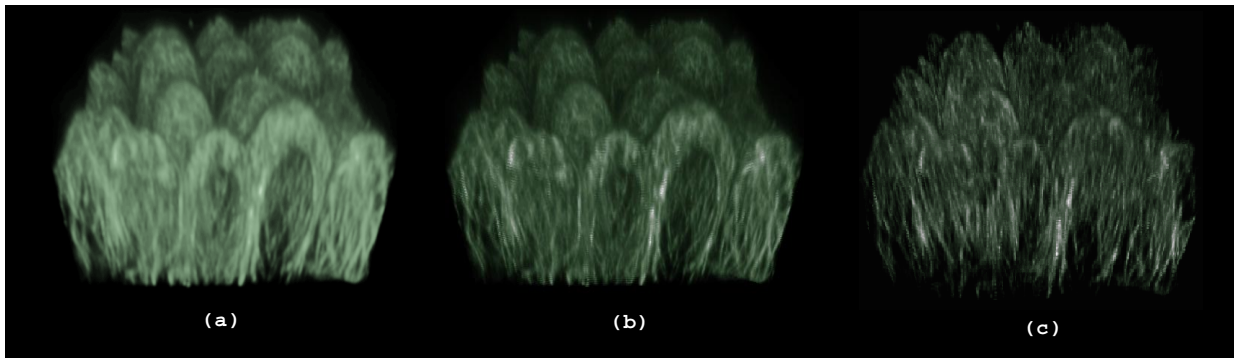


Figure 4: (a) Volume rendering by a linear ramp transfer function; (b) After a median filtering; (c) Median filtering followed by a highpass filter with a Laplacian mask

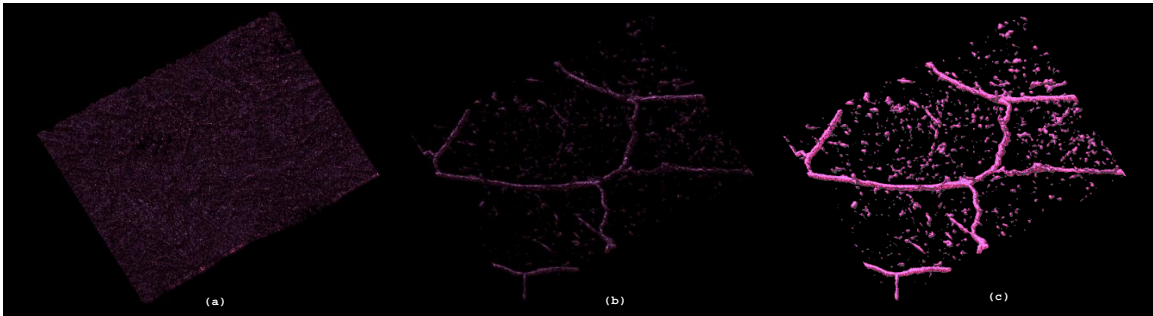


Figure 5: (a) Volume rendering by a linear ramp transfer function; (b) *Laplacian* masking followed by *unsharp masking* with $\gamma = 3$; (c) *Laplacian* masking followed by *unsharp masking* with $\gamma = 10$

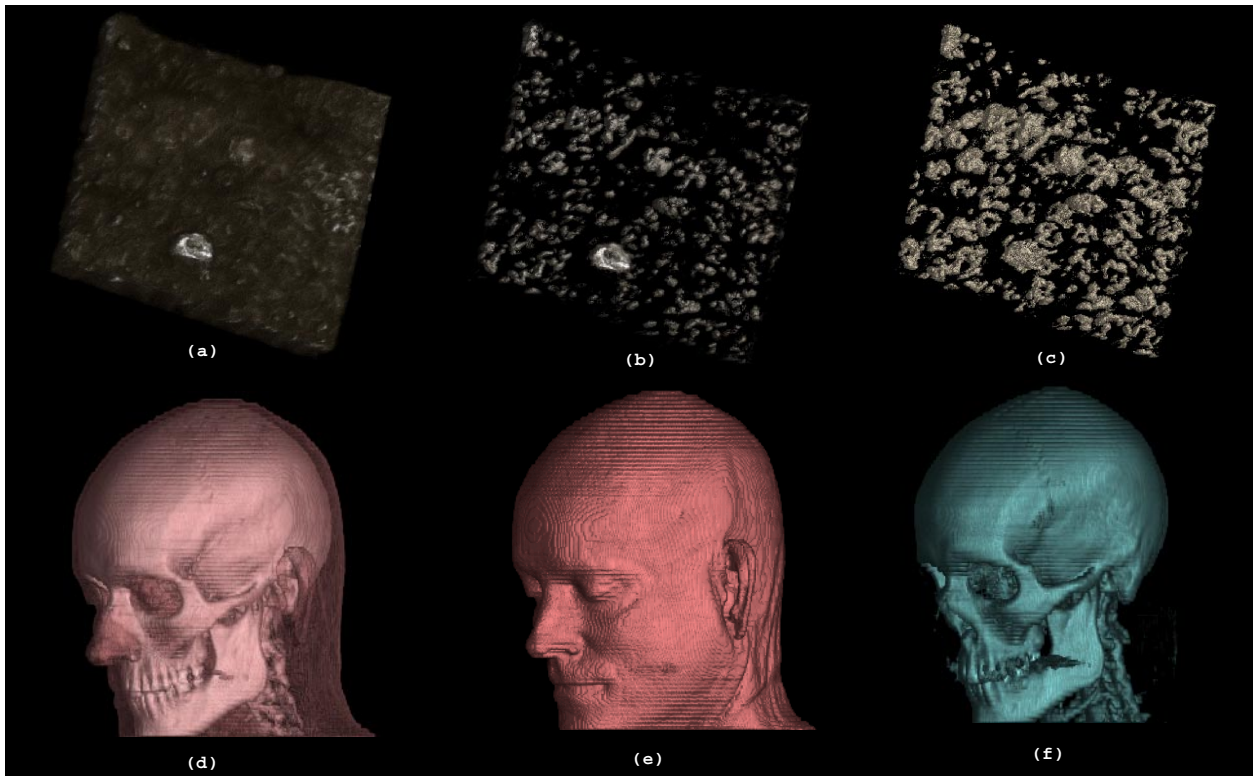


Figure 6: Surface rendering examples. (a) Golgi complex rendered by a linear ramp; (b) Golgi surface rendering with gradient threshold 60; (c) Golgi surface rendering with intensity interval [77, 97]; (d) CT head surface rendering with gradient threshold 40; (e) Skin surface rendering with intensity interval [53, 69]; (f) Skull surface rendering with intensity interval [74, 94]

- [13] Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990.
- [14] Azriel Rosenfeld and Avinash Kak. *Digital Picture Processing*. Academic Press, 1982.
- [15] T. Schiemann, M. Bomans, Ulf Tiede, and H. Hohne. Interactive 3D-segmentation. In *Proc. Visualization in Biomedical Computing, Chapel Hill, NC*, 1992.
- [16] Craig Upson and Michael Keeler. V-buffer: Visible volume rendering. *Computer Graphics, SIGGRAPH'88*, 22(4):59–64, August 1988.