Volume Visualization of Multicolor Laser Confocal Microscope data

Anshuman Razdan[1], Kamal Patel[2], Gerald E. Farin[3] and David G. Capco[4]

Abstract:

The objective of this work is to give biologists the ability to visualize *Multicolor Laser Confocal Microscope* (MLCM) data sets. We present the application of volume ray casting techniques to create fast and accurate visualization of the spatial structures in MLCM data sets. We provide various volume pre-processing routines for noise removal, smoothing and edge enhancement. The tool has an interactive interface to assist the operator in classifying the volume using various transparency mappings. We have developed volume rendering using volumetric ray casting based upon a perspective camera model. We illustrate an innovative way to simultaneously visualize LCM data set from three lasers using voxel level merging. A method to generate key frame animation sequence of an arbitrary fly through path is also included in the visualization tool.

Keywords: Volume visualization, confocal Microscopy, ray casting and computer graphics.

[1] Director PRISM, MC 5106, Arizona State University, Tempe AZ 85287, (480) 965-5368, Razdan@asu.edu.
[2] Senior Software Engineer, Altera Corporation, San Jose, CA, kpatel@altera.com
[3] Professor, Computer Science and Engieering, Arizona State University, Tempe AZ 85287, farin@asu.edu.
[4] Professor, Biology, Arizona State University, Tempe AZ 85287, dcapco@asu.edu

1 Introduction

The goal of this work is to produce an interactive tool for visualization of Multicolor Laser Confocal Microscope (LCM) data sets. Image processing and analysis methods are applied to these data sets and high-resolution 3D information about the structures of interest was generated. The term “Multicolor” is used in context with Confocal Microscope[1]. The term “tool” implies that the operator operates (sets the parameters and analyzes the results) with an interface. This type of visualization tool has two desired goals:

?? The tool must provide the operator with an interface that is easy to learn and use, and,
?? The operator must be able to select/classify the features of interests and visualize the results with relative speed and ease.

TINVIZ solves the first by giving the operator an interactive mouse driven interface written in OpenGL. In addition to being easy to learn, the interaction process gives the operator continuous visual feedback. As for the second, TINVIZ provides a fast perspective ray casting engine and an interactive classification tool.

1.1 Multicolor LCM

Many biological problems cannot be unambiguously characterized by the examination of a single parameter, hence techniques for independent detection of signals from multiple fluorescent probes are required. At present, this can most easily be done by using fluorescent probes that can be viewed selectively based on differences in their excitation and emission spectra. This is known as Multicolor Laser Confocal Microscopy [1]. It is used to acquire monochromatic images of several fluorophores in different spectral regions. Here, because of independent detection of signals from each fluorophore, each image contains information on the localization of staining of different substances for the same field of view.

The scale of objects evaluated in confocal microscopy is measured in microns (?). 1? = 10^{-6} meters. Lasers may penetrate an object from 20 - 120 microns. A rather crude scan would have 5 microns between steps. Different lasers provide "coherent light" at different wavelengths, so with different lasers one can reveal different features within an object.

1.1.1 Data Characteristics

LCM data typically comes as a series of 2D grayscale images representing optical sections through a specimen. These serial images are commonly referred to as the image stack. All of the

images in the image stack in most cases are aligned along the X, Y and Z-axes. Typical image sizes are 256*256, 512*512 and 768*512. The selection of the image size is usually dependent on desired size of the final 3D data set (the series of 2D images collected). For example, an image stack consisting of 50, 512*512 8-bit images occupies 12.8 MB of storage space. Data generally has poor contrast, poor Signal-to-Noise (SN) ratio and low intensity gradients, and the image stack may have unequal resolutions in spatial and lateral directions. For example, in some of the data sets used, two adjacent pixels in an image are 0.16? apart while two consecutive images are 0.5? apart.

1.1.2 Motivation

Almost all of the existing LCM data visualization systems are based on visualization of a single data set at a time. The techniques for the display and analysis of the multiple data sets collected by multicolor laser confocal microscopes are still being developed. While many options currently exist for two-dimensional reconstruction and visualization, there are no tools that we know of that permit the three-dimensional visualization of such data sets. The primary contribution of this

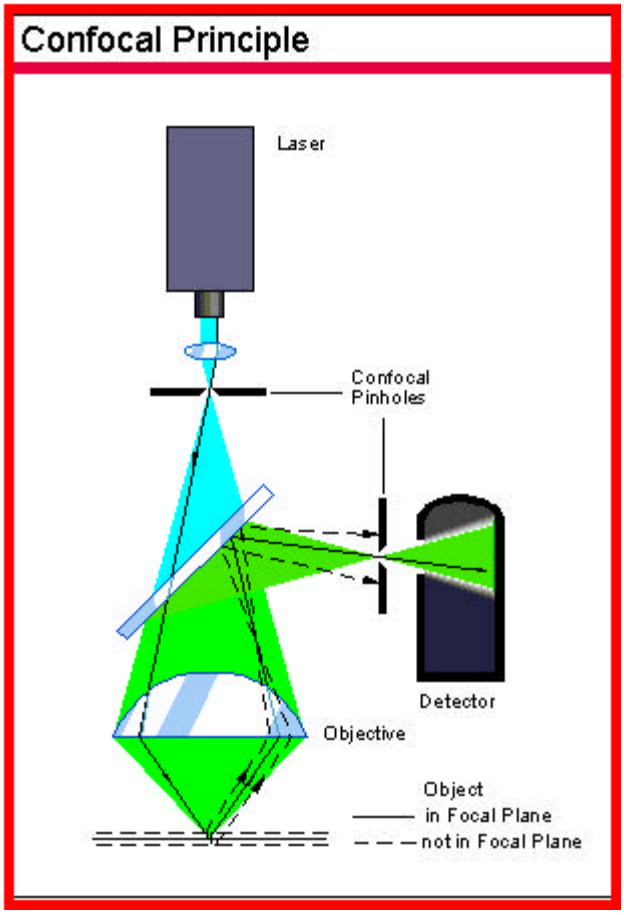

Figure 1. Illustration showing the data collection by a microscope using Confocal principle.

research is an innovative way to visualize more than one LCM data set at a time.

1.2 Volume Visualization

With the extensive research and development effort for sampled 3D volumetric data sets, volumetric visualizations are becoming increasingly important. Everyday new applications of volume visualization are emerging. There are two forms of volume visualization: Surface rendering, and Direct Volume Rendering.

1.2.1 Isosurface Rendering

Isosurface (surface of same density) rendering is an indirect method of obtaining an image from volume data. In isosurface rendering, the volume is represented by geometric primitives. For a given isosurface value, the surface is detected at each voxel by applying a binary selection procedure to that voxel. Geometric primitives like triangles are then fitted to the detected surface. At last those primitives are shaded and rendered. These include Cuberilles by Herman and Lui [2] and Marching Cubes by Lorenson and Cline [3].

There are few advantages of surface extraction methods. The time consuming processing step is done only once and the resulting primitives can take advantage of standard 3D graphics acceleration hardware which is very useful for navigating through data containing fairly well defined surfaces. Iso-density surfaces are good from modeling point of view as they are represented by well defined geometric primitives.

The disadvantages include loss of accuracy when visualizing small or fuzzy structures with densities that are not near the isosurface. The information on the interior of the isosurface is also lost. Marching cubes algorithm cannot accurately extract non-manifolds. A very large number of primitives are needed to map the isosurface. Visual aliasing appears because of binary selection, and one has to know what to look for the optimum iso-value for selection.

1.2.2 Direct Volume Rendering

Direct volume rendering does not involve the intermediate representation of volume data to geometric primitives rather the volume is directly visualized by projecting the data onto an image screen. Vannier, Marsh and Warren [4] proposed the first direct volume-rendering algorithm based on depth shading. Volume ray casting is a well-understood technique; it generates good quality images by compositing sampled information like intensity and gradient at discrete locations inside the volume data. Levoy [5] proposed one of the most popular algorithms for

volume rendering. Westover [6] suggested the Splatting algorithm that is basically another variation of the ray casting algorithm.

Direct volume rendering is structure independent. Even small and fuzzy structures can be visualized more accurately than using surface rendering. There is no need for intermediate geometric representation of the volume data set, and interior information is not thrown away. However, sometimes cloudy interiors are hard to interpret. Direct volume rendering is slow because it is computationally intensive, and aliasing appears in the final image.

Most of the present fast rendering methods work on volume pre-shading, pre-classification and caching of normals. This pre-calculation and caching is generally suitable for static rendered views and when the data set size is not too large. For volume navigation it is better to have dynamic shading based on viewer's eye. Also, for large data sets it is not feasible to cache surface normals as it may require a lot of runtime storage.

The direct advantage of a ray casting based visualization tool is the ability to visualize the heterogeneous and very weak inner structure of cells, which cannot be achieved through surface graphics. Also, the methods based on Splatting are not very useful because of the requirement of visualizing more then one data set at a time. Splats project a whole voxel at a time, and in this case it is very hard to preserve depth information, which is very crucial while merging more than one volumetric data set.

2 Volume Rendering

In this section, the volume rendering technique based upon the ray casting algorithm is discussed. The visualization pipeline and the ray casting algorithm is explained. The mathematical foundation for the volume rendering integral is given in[7].

2.1 Visualization Pipeline

A series of transformations are carried out on the data in order to convert the original data into an image, which can be displayed. The original model proposed by Haber and McNabb [8] divides the visualization process into three main transformations: data enrichment and enhancement, visualization mapping, and

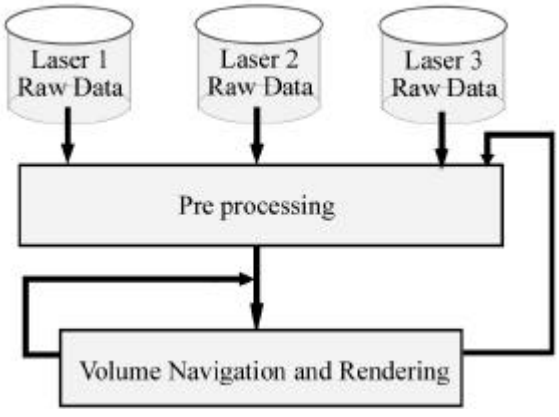

Figure 2: The Visualization Pipeline

rendering. However, we absorbed the visualization mapping into pre-processing for data enrichment and enhancement, and rendering.

The visualization pipeline used here can be represented as a simple flow chart shown in Figure 2. The inputs to the pipeline are the raw volumetric data sets, collected from the three lasers of the confocal microscope. The visualization here is a two-stage process. In the first stage, raw volumes are pre-processed and in the second stage, the user performs interactive volume navigation.

2.1.1 Data Enrichment and Enhancement

In this step, we take the raw data and alter it into a format which can act as input for the required visualization operations. This might involve interpolation of results to obtain additional results, or the filtering of noise from the system. Once the data has been translated into a useable format it is used to construct an abstract object called an Abstract Visualization Object (AVO). AVO is used to represent the data, which is being modeled, and can contain several attributes such as color and opacity. Figure 3 shows the three phases of the pre-processing stage. These are data enhancement, classification and optimization encoding.

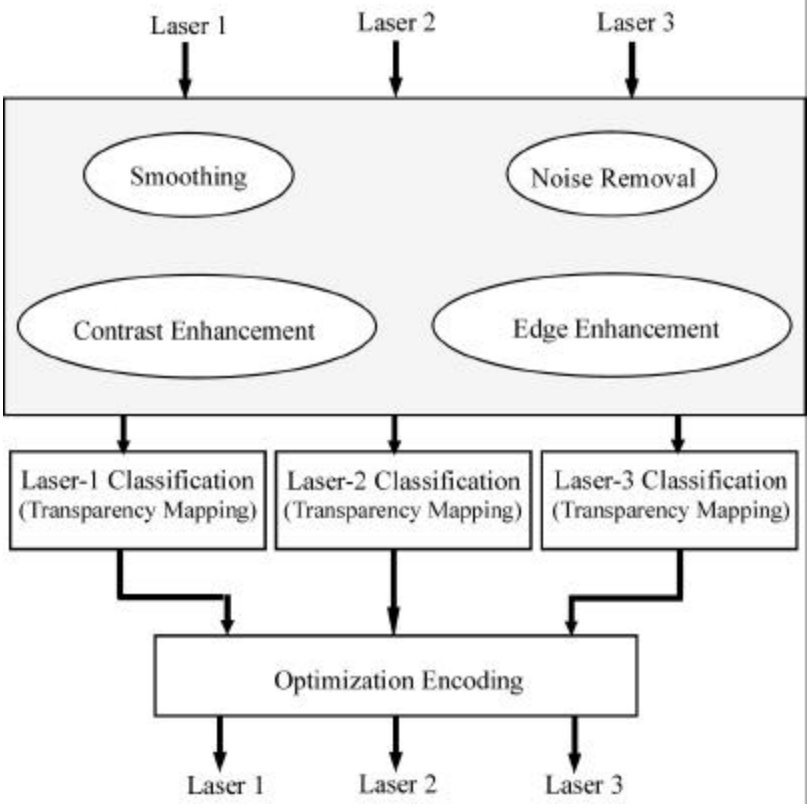

Figure 3: Pre-processing stage of multicolor LCM Data

2.1.2 Rendering

The final stage in the visualization pipeline involves displaying, or rendering, the object on the screen. This stage will make use of ray casting to transform the AVO into a displayable image. The volume navigation and ray casting stage contains two major phases: viewing parameter setting, and ray casting. Figure 4 shows the volume navigation stage.

Various viewing parameters can be set through the GUI. Ray casting is then used to generate volume rendered views. As TINVIZ deals with at most three lasers, there are total four images to be computed, one for each separate laser and one for the combined result.

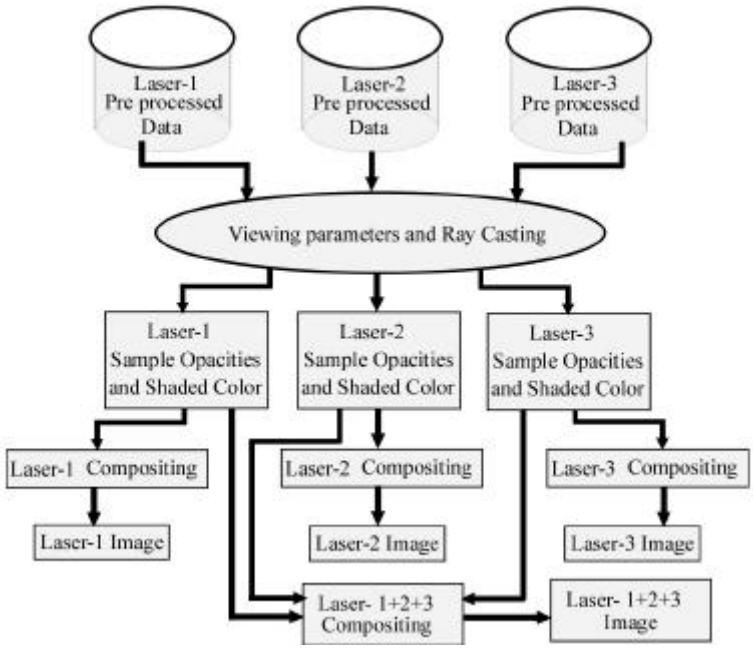

Figure 4: Volume rendering and navigation stage

2.2 The Volume Rendering Equation

Volume rendering is an approximate simulation of the light propagation through the volumetric data sets. If the volumetric data is considered as a translucent gel, the amount of light attenuated by an object is directly proportional to incoming intensity. Also the object itself emits light of some intensity. This may not be the most accurate simulation of the real physics, but it gives better results for data visualization.

Here, only front to back ray casting is considered. Figure 5 shows the basic notations used [7]. The front-to-back compositing can be written as:

$F = I_n$

$? = 1.0$

For i = n-1, n-2, … , 0

$? = ? \, t_{i+1}$

$F = F + ? \, t_i$

Endfor

where:

i = variable used to denote the samples increasing towards eye,

n = total number of samples along a ray,

F_i = intensity value from eye till the i^{th} sample,

I_n = Intensity value at eye,

t_i = transparency value at the i^{th} sample and

? = Accumulated transparency

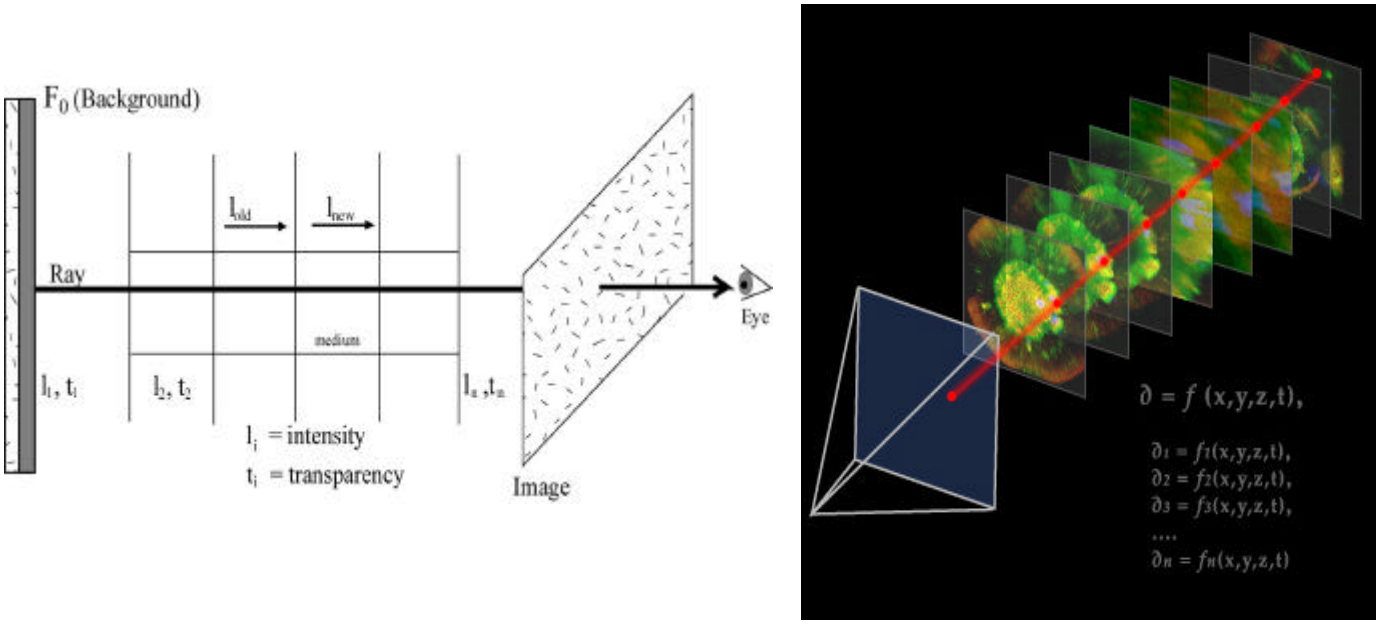

Figure 5: Two illustrations showing the ray casting algorithm

The monotonically decreasing variable ? is used to hold accumulated transparency. If there is a sequence of objects with intensity I_i, and transparency t_i then the intermediate intensities are calculated as:

$$F_{i+1} = t_{i+1} F_i + I_{i+1,\, i} = 0, 1, \ldots, n \quad (1)$$

which is the same as:

$$F_n \; ? \; \sum_{i=0}^{n} I_i \prod_{j=i+1}^{n} t_j \quad (2)$$

it can be written that,

$F_{i+1} - F_i = -(1- t_{i+1}) F_i + I_{i+1} = -?_{i+1}(F_I - C_{i+1})$

where: ? = (1-t) = opacity, C = Color value. and I = ? C

Using this notation, equation (2) can be rewritten as:

$$F_n \; ? \; \sum_{i=0}^{n} (1 \; ? \; t_i) C_i \prod_{j=i+1}^{n} t_j \quad (3)$$

If sequences are replaced with functions using the notation,

$F_i = F(x), \quad F_{i+1} = F(x+?x), \quad ?_{i+1} = ?(x+?x), \text{ and } C_{i+1} = C(x+?x),$ (4)

then

$$\frac{F(x?\,?\,x)?F(x)}{?\,x}\,?\,\frac{?(x?\,?\,x)}{?\,x}(F(x)?C(x?\,?\,x)) \quad (5)$$

For each image pixel, a ray is cast through the volume. At evenly spaced intervals along the path of the ray, opacity and color are computed in the volume. Equation (4) is used for compositing the colors and opacities along the ray.

The above-described algorithm is followed for each of the three laser volumes for multicolor volume rendering. The computed intensities from each laser volume are combined to obtain a combined intensity value using the voxel level merging. The voxel level merging is described in detail in section 3.5.6.

3 Implementation

In this section, we describe the implementation of the opacity based ray casting algorithm. The modifications made to make it suitable for volume rendering of three volumetric data sets at a time are also discussed. Sections 3.1 to 3.4 discuss the pre-processing and, 3.5 and 3.6 discuss the rendering as described in section 2.

3.1 Data structures

3.1.1 Data structures for Per Laser Data

The volumetric data set as a whole is composed of three different volumetric data sets, one for each laser in the system. A data structure (**C_LaserData**) represents each laser in the system. C_LaserData holds most of the information about volumetric samples for that laser in the form of different data structures. This information is as follows**:**

?? Volume data,

?? Homogeneous Voxel flag,

?? Histogram,

?? Transparency Look Up Table, and

?? Image generated by Ray Casting.

Three-dimensional arrays of unsigned char are used to store volume data, as the volume elements are 8 bits per sample. The 3D array is allocated as a 1D sequential stream of Nx.Ny.Nz bytes,

where Nx = Number of samples in X direction, Ny = Number of samples in Y direction, and Nz = Number of samples in Z direction. Here, voxel values are stored according to the principal volume axis, i.e. for each XY location there is a stream of Nz bytes. The 3D array allows efficient traversal of data volume and random access to voxels.

For each voxel that is homogeneous (the values at all vertices of the voxel are the same), a one bit flag is set. Homogeneous voxel flags for the whole volume are stored in a 1D stream of (Nx - 1)*(Ny - 1)*(Nz - 1) / 8 bytes. These flags are set during preprocessing step and tested to check for homogeneous voxels during the ray casting step.

Another data structure (C_Image) represents the composite ray cast image for each laser. The C_Image class stores image width, height and 24 bit color value for each pixel, where the total number of pixels is width*height. Image data is represented by 1D array of pixels. Each pixel is a collection of the three 8 bit values: red, green and blue component of the color.

3.1.2 Data structures Common to all Laser Data

Two main data structures that are common to all laser volumes are empty space distance transform and viewing camera.

Empty space distance is the distance that can be skipped during the ray casting step to avoid computation in background/empty voxels without affecting the end result. These distances are calculated during the preprocessing step. They are stored as a separate 3D array of *unsigned chars*. As the volume elements of this class are 8 bits in size, the largest value that can be stored is 255. This value serves as the radius of a sphere within which all voxels are background/empty. It is important to note that this data structure is view independent.

3.1.3 Ray Template

The ray template is used to advance the ray during ray casting. The new point on the ray can be easily computed by adding a fixed offset from the current position. The template for a ray is represented by combination of the starting point and ray direction vector. Thus the point on the ray at a distance ? x can be calculated as:

XYZ(? x) = starting_point[ray] + ? x.direction[ray] (6)

3.2 Cache Coherent Addressing Scheme

3.2.1 Cache Coherency

The 1D array representation of the volume data is appropriate when volume data is small. For larger volumes such a representation may result in a poor performance of the system due to its cache non-coherent nature. If a ray traverses the volume in the same direction as the direction of physical memory storage, then each time a new voxel access is made, it is already in the cache as cache loads data as memory pages. But as the ray traverses the volume in the direction other than the memory storage direction, it may have to reload the new memory page in the cache each time a new voxel access is made depending upon the memory page size. This usually results in heavy bus traffic and increase in rendering time.

The cache loading algorithms are mostly LRU (Least Recently Used), so they swap out the least recently used cache-memory page. This means that the LRU algorithm will try to keep voxels in the cache memory that are in the neighboring region of the current position.

The remedy to this problem is to linearize access to voxels. This can be achieved either by object order linearized traversal of volume or by changing the addressing scheme of the 1D array of voxels to put neighboring voxels into the same memory page.

The first solution results in a splatting like algorithm for volume rendering. Since ray casting instead of splatting is used for volume rendering in TINVIZ, the second solution is obvious. Splatting needs visibility sorting of voxels before compositing them. The voxel influenced (covered) pixels determination is computationally expensive, and hence second choice was selected for TINVIZ. The solution is as described below.

3.2.2 Addressing Scheme

This development is due to Sakas, Grimm, and Savopoulos[9]. Here, volume data is represented by small sub cubes of Caddr.Caddr.Caddr size. The Caddr (size of one side of the cube) is chosen in such a way that all volume elements in this subcube can fit in the same cache memory page. Typically the cache page sizes are 512, 1024 or 2048 bytes. Thus a better candidate for Caddr is a value between 8 and 12.

Now according to this scheme, the address of a volume element at (i, j, k) location can be calculated as follows:

Address (i,j,k) = base_address + Gaddr[i] + Haddr[j] + Iaddr[k]

where:

base_address = starting address of the 1D array,

Gaddr[i] = (i / Caddr)*$Caddr^3$ + (i % Caddr),

Haddr[j] = (j / Caddr)* (Nx / Caddr)*$Caddr^3$ + (j % Caddr)*Caddr,

Iaddr[k] = (k / Caddr)* (Nx / Caddr).(Ny / Caddr)*$Caddr^3$ + (k % Caddr)*$Caddr^2$. (7)

Here, Gaddr, Haddr and Iaddr are address look up tables of the sizes Nx, Ny, and Nz respectively. They are pre-calculated only as the initialization step.

3.3 Volume Preprocessing

TINVIZ tries to improve the data set quality by means of filtering. Various image-processing filters are applied on slice-by-slice basis.

3.3.1 Noise Reduction

A Median filter is used to eliminate noise and background artifacts and to smooth sharp edges. These filters have the general affect of smoothing images, but also tend to remove some of the detail in small objects. The median filter has been proven a good tool, since it effectively reduces noise and small surface artifacts, while maintaining the sharpness of contour [10]. Here the input pixel is replaced by the median of the pixels contained in a window around the pixel at (m,n), that is

V(m, n) = median{y(m-k, n-l), (k, l) ? W} (8)

where:

W is a suitably chosen window,

V is the output image and

y is the input image.

3.3.2 Contrast Enhancement

The contrast and brightness of the image stack can be adjusted to enhance perception of the sampled specimen. Usually this can be achieved by changing the ramping of the gray scale values for the data set usually does this. The contrast enhancement is implemented using Contrast Stretching and Histogram Equalization. These filters are discussed in more detail in [11]. In contrast stretching the slope of the gray level histogram is changed to make it occupy the whole gray level range (0 – 255). This improves the overall visibility of the image. Histogram

equalization can be used to improve contrast by a non-linear mapping of the gray levels in an image. This technique is most commonly used when the gray levels are concentrated in a small portion of the range of possible values.

3.4 Volume Classification

Volume classification here refers to the mapping of volumetric data values to transparency (opacity). Both the thresholding and the transparency mapping are provided for classification. Thresholding gives complete binary selection while the transparency mapping can be used to achieve non-binary classification. This type of classification is particularly important for LCM data sets, where because of the unknown internal structure; the complete binary classification is impossible.

The classification is implemented in the form of a look up table. As the volume data values vary between 0 and 255, the size of the look up table is 256. This look up table is used to find the transparency for the vertices of a voxel. Transparency within the voxel is calculated using tri-linear interpolation.

For threshold and transparency mapping a functional Bezier curve based interactive mapping tool was developed. In a display window, a graph of data values (0 to 255) and corresponding transparency is plotted (Figure 6). The graph is a quadratic Bezier curve with three control points. The range of this curve is the transparency value between 0 and 1. There is a Bezier-Curve for each laser in the system. Each curve is drawn using the color assigned with that laser data volume. The Bezier theory of curves is discussed in detail in [12].

3.4.1 Bezier Development

The transparency curve is given by

$$T^n(t) = \sum_{i=0}^{n} control_i \cdot B_i^n(t) \tag{9}$$

Where, B^n_i is a Bernstein polynomial of degree n and is given by

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{(n-i)} \tag{10}$$

where:
number of control points = degree of the curve + 1

Since, this is a functional case ($y_i = f(x_i)$) , the arrangement of control points is restricted as points, to have (x_i , y_i) coordinates and $x_i <= x_{i+1.}$

One can interactively add or remove control points of the curve. This changes the degree of the Bezier curve, and makes it easier to introduce sharp discontinuities in the curve. The operator can modify the curve to achieve the desired classification. Interactive thresholding of volume data can also be performed with this tool by setting the transparency value to maximum (here 1.).

Such a tool allows the user to perform fast and efficient classification. This is particularly useful when the user does not have any well-defined classification as is available for some CT/MRI data sets compared to the data from the confocal microscope. Adjusting the curve gives smooth transition between consecutive transparency values. This reduces aliasing artifacts. Sharp peaks/valleys in the transparency curve result in isosurfaces i.e. assigning the transparency of the desired iso-value 0 and for rest of the gray levels making transparency 1, makes the isosurfaces at that iso-value visible.

3.5 Ray Casting

The algorithm discussed in section 2 is followed to implement the multi laser ray casting. For each pixel in the image plane, a ray is shot and the intensity values are composited along that ray. The basic process, which is repeated for each pixel, can be divided into four segments: Ray advancing, Interpolation, Shading, and Compositing. This process is the heart of the ray casting engine.

3.5.1 Ray Advancing

Ray templates are implemented for ray advancing. The ray template stores the ray direction and the starting point of the ray. The step size to advance the ray is fixed (by the user), i.e., the samples are taken in the volume at uniform intervals.

3.5.2 Interpolation

After having the voxel and the parameter values, there is a need to find the volume data values at the sample location for each laser. Trilinear interpolation is used to calculate the data values. After having interpolated the data values, the transparency mapping of each laser is used to find the transparency at the sample location. Trilinear interpolation is given by the following formula:

$$F(x, y, z) = v0 * (1 – x) * (1 – y) * (1 – z) + v1 * (x) * (1 – y) * (1 – z) +$$
$$v2 * (1 – x) * (y) * (1 – z) \quad + v3 * (1 - x) * (1 – y) * (z) +$$

$$v4 * (x) * (1 - y) * (z) \quad + v5 * (1 - x) * (y) * (z) +$$
$$v6 * (x) * (y) * (1 - z) \quad + v7 * (x) * (y) * (z) \qquad (11)$$

where:
F(x, y, z) is a function that is to be reconstructed inside the voxel and
(x, y, z) are the parameter values inside the voxel.

3.5.3 Shading and Lighting

According to the volume-rendering pipeline discussed in this work, the view dependent shading is required at a particular sample location inside the volume. The shading helps enhance the realism but does not participate in the volume classification.

The Phong Shading model[5] was used to provide satisfactory and smooth shading of the volume data sets. This illumination model computes the color at a point, in terms of light directly emitted by light sources, and of ambient light. It takes into account specular reflections.

A lighting model is used with a one-point source of white light. There are two possible choices for the lighting model Fixed Lights with pre shaded volume, and Moving Light (Miner's Lamp) with shading on the fly.

As interactive volume navigation is one of the primary goals of TINVIZ, the first choice is inappropriate as in this case some areas do not receive light at all and hence are hard to see. If more light sources are used to remedy this problem, it reduces the shape realism because of uniform light from all directions. Hence, the second choice. The light source is located at the same location as that of the viewer's eye. It moves along with the viewer to give the "Miner's Lamp" like effect.

The light rays traverse the volume in the direction of the viewing rays. The lighting model uses information derived from the gradient computation to estimate the normals.

3.5.4 Fast Normal Estimation

Shading on the fly requires the calculation of the normal. Generally the normal at a certain point is

$$N(x,y,z) ? \frac{? f(x,y,z)}{|? f(x,y,z)|} \qquad (12)$$

where:

? f(x, y, z) is the gradient at the location (x, y, z).

The gradient at each voxel vertex is approximated by the Center Difference method as follows:

? f(x, y, z) = (f(x+1, y, z) - f(x-1, y, z), f(x, y+1, z) - f(x, y-1, z), f(x, y, z+1) - f(x, y, z-1)) (13)

The gradient everywhere inside the voxel is approximated by trilinear interpolation of the gradients at voxel vertices. This requires three Center Differences and three tri-linear interpolations. This is too expensive to compute on the fly. One remedy to the problem is to pre-calculate and cache (store) gradients at each voxel vertices. However, this may cause a significant memory requirement, almost three times the memory used to hold actual volume data. In TINVIZ more than one-volume data sets are to be rendered; therefore, it is too expensive to store the pre-computed gradients. Even if enough memory is provided, the three trilinear interpolations will cost twenty-four multiplications, which are too expensive for interactive rendering. Therefore, a fast analytic gradient method is used for gradient approximation.

Partial derivatives of the trilinear equation with respect to x, y, and z can be defined to calculate the analytic gradient. Thus the gradient at (x, y, z) location within a voxel V is given by:

? f(x, y, z) = (s, t, u)

where:

s = (1 – y) * (v1 – v0 + z * (v4 – v3 + v0 – v1)) + (y) * (v6 – v2 + z * (v7 – v5 + v2 – v6))

t = (1 – x) * (v2 – v0 + z * (v5 – v3 + v0 – v2)) + (x) * (v6 – v1 + z * (v7 – v4 + v1 – v6))

u = (1 – x) * (v3 – v0 + y * (v5 – v2 + v0 – v3)) + (x) * (v4 – v1 + y * (v7 – v6 + v1 – v4)) (14)

s, t and u are the partial derivatives of the trilinear function F with respect to x, y and z respectively.

The analytic gradient scheme costs twelve multiplications. More importantly, if the voxel is homogeneous, there is no need to calculate the gradient. This scheme works well at tolerable image degradation.

3.5.5 Volume Navigation

Volume navigation involves moving the image plane in a defined path around the data set. This allows the data to be viewed from a slightly different direction each time. If the frames, or individual images of the picture, can be viewed quickly enough then the effect of motion parallax will give the image the illusion of 3 dimensions. The volumetric data sets from each laser, acts as a “virtual environment”, representing the internal structure. TINVIZ provides the means to

navigate through this environment. During navigation, TINVIZ generates volume-rendered views at various viewing sites along the navigation path. This is done for each laser in the system. Thus the operator is virtually navigating in four different worlds at the same time, one from each laser volume data and one that combines the three laser volume data sets.

The Cartesian coordinate system is used for both the data coordinates and camera coordinates. The viewing pyramid is defined in the data coordinate system. The image plane, from which the rays are cast, forms the base of the viewing pyramid. The viewer's eye is at the top vertex of the viewing pyramid. The distance between eye and the image-plane (eye distance) is the height of the viewing pyramid. The eye distance defines the view angle. The direction from the eye to the center of image plane is the viewing direction. See Figure 6. The light source is placed at the eye's location. The image stack that forms the data volume is fixed in the coordinate system. On the other hand, the orientation and the position of the viewing pyramid can be modified. The volume navigation is performed by incremental movements or translations and rotations of the viewing pyramid. The movements and rotations are implemented as standard computer graphics translation and rotation matrices. Each time the orientation of the viewing pyramid is modified, the ray directions for each pixel are calculated and stored in a two dimensional grid.

In a typical situation, the viewing pyramid is very small with respect to the data cube and it lies in side the data cube. A navigation path is traced out as the user travels. At each step of the navigation path, the user sees a 3D view of the local environment around that viewing site. At each viewing location ray pyramid is used to generate the rendered view.

3.5.6 Compositing

In the compositing stage, the intensity of the light at New_Sample[i] is calculated and composited with the previously composite samples. For each laser l,

Intensity[l] = color[l]*opacity[l]
Composite[l] = Composite[l] + Intensity[l]*Tau[l]
Tau[l] = Tau[l]*transparency[l] (15)

where: Tau[l] is the cumulative transparency of laser l.
Combined = Combined + Intensity[l]*Tau[l]*Percentage[l], l = 0, 1, 2 (16)

where:
Combined is the combined composited image from more than one laser and
Percentage[l] is the percentage of material from laser l present in voxel (a, b, c)

Following this process, a pixel color is obtained on each of the image planes and also on the combined image plane. When the ray exits the volume boundary, it hits the background immediately. So for each laser l,

Composite[l] = Composite[l] + Back_ground*Tau[l] (17)

And for the combined image,

Combined = Combined + Back_Ground*Min_Tau (18)

where:

Min_Tau is the minimum Tau values from all of the lasers.

The Tau for each laser is initialized to 1.0, and the Composite and Combined values are initialized to complete black (0, 0, 0) color. The most time consuming part is the shading. There are a lot of optimizations possible to improve this per ray process. Those are discussed in the next section.

3.5.7 Multi Laser Ray Casting

Ideally for *n* number of volume data sets, one needs to do ray casting n times and it will cost n times the cost for the ray casting in a single laser data set. However, there are certain calculations that are common to all laser volume data sets such as ray advancing using ray templates, voxel identification, and trilinear parameter calculation.

The ray traverses in all volume data sets at the same time on the same path. The sample location (the co-ordinate in 3D data space (x, y, z)) is the same for all data sets in side a volume. The voxel in which the (x, y, z) sample is located is the same and the trilinear parameters inside that voxel remain the same for each data set.

The transparency and color accumulation needs to be calculated separately for each laser data set.

3.6 Optimizing Rendering Speed

In TINVIZ, the ray casting process is optimized using few existing optimization techniques. For faster memory access the cache coherent addressing is used as discussed in section 3.2. Image Space Coherence by Levoy [8] was used for an adaptive ray casting algorithm that exploits the coherency and band limited nature of volume data so that on average, less than one ray per pixel is needed to produce an image that is visually comparable to that obtained by conventional ray casting. While using Object Space Coherence, if the current voxel is homogeneous, the function value inside the voxel is constant and there is no need to do trilinear interpolation. The empty space in multi-laser laser data sets is the region in the data sets where there is just background

voxels in all the lasers. A 3D-distance transform is used to accelerate the rays through empty space. The Early Ray Termination was used to terminate the ray when the ray becomes more opaque and hence, less sensitive to errors in coloring.

4 Conclusion

Volume rendering using ray casting has proved to be an effective technique for visualizing multicolor LCM data sets. As shown in the color plates (Figures 6-9), it can generate good

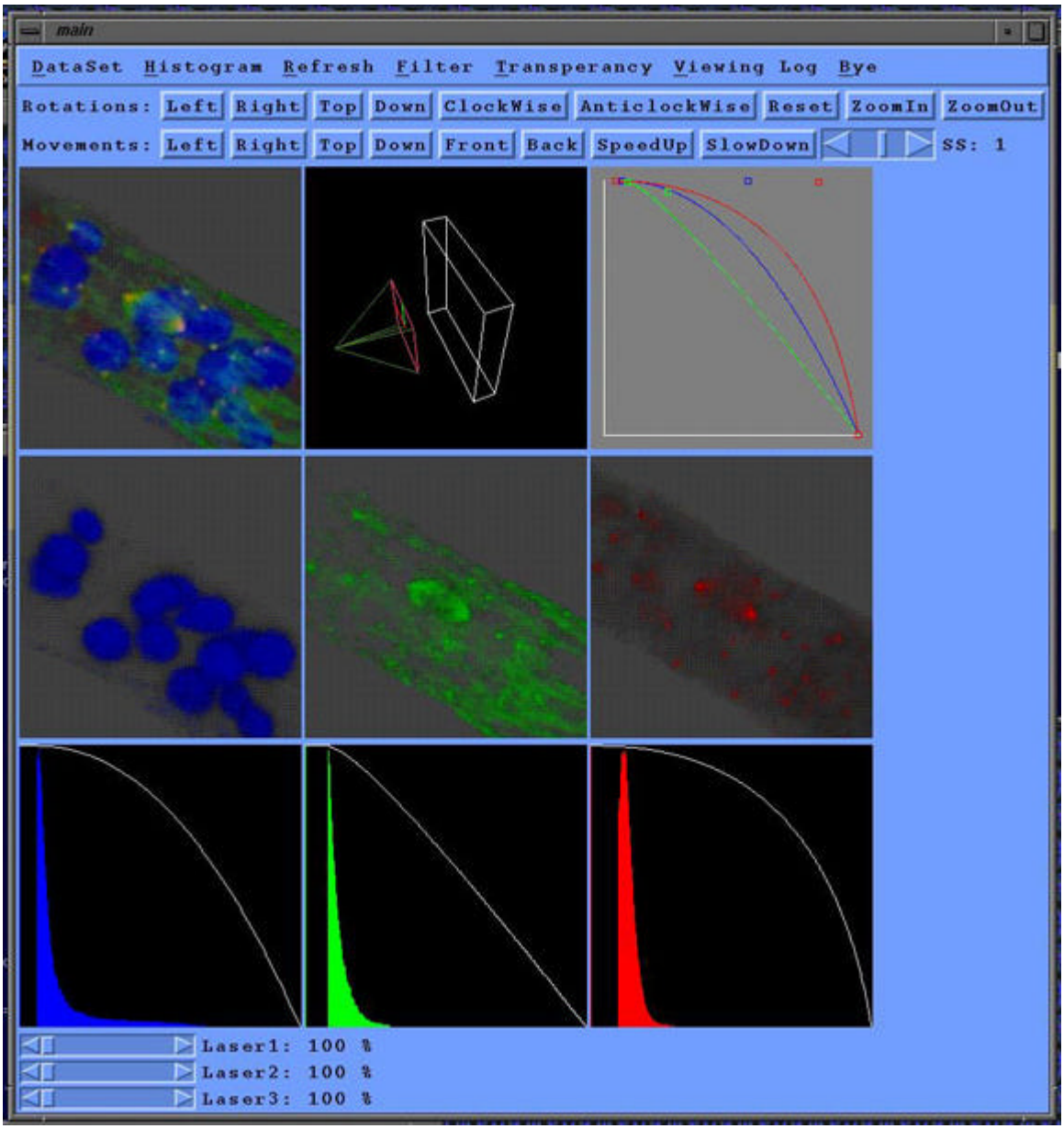

Figure 6: A snapshot of the TINVIZ tool. The bottom three windows show the Bezier tool on the histogram of each laser data, the middle windows show volume visualization of each laser data, and the topleft is the composited image. The top middle shows the position of viewer w.r.t. to the data volume. The top right window shown the opacity curves as a result of the application of Bezier tool.

quality images, which may be sequenced together to generate an animated sequence.

TINVIZ supports data pre-processing, fast classification and volume navigation. An important issue was to handle multicolor confocal data sets and simultaneously visualize more than one

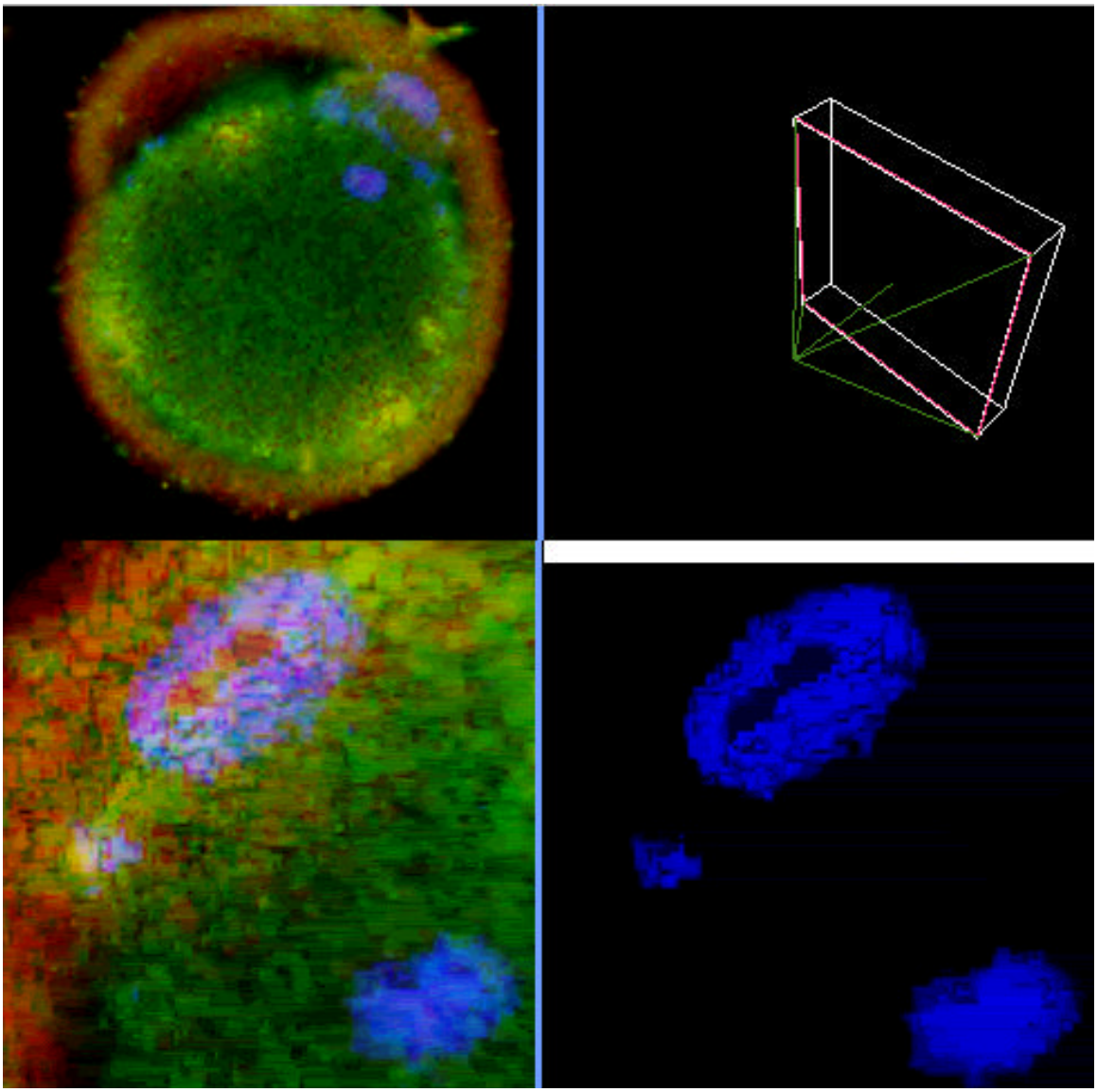

Figure 7: Screen shots showing (a) image of an egg cell looking from outside (topleft) (b) viewing volume with apex of the green pyramid as the user position (top right) (c) a particular view of the cell where green is MAP Kinase, red is Cyclin B1 and the blue is the DNA (bottom left) and, (d) donut shape of DNA emerging as a result of 3D volume visualization which the cell biologist would have been unable to detect before.

volumetric data set at a time. Equally important was the problem to accelerate the ray casting

process. The capability of visualizing more than one LCM volumes at a time allows the insight in a process of great biological importance not completely understood yet.

5 Future Work

In order to make use of motion parallax with real time visualization, images must be produced at the rate of at least 10 frames per second. As the time required to produce the images is in the order of tens of seconds, the real time visualization of volume data sets is still an active area of research. The biggest challenge for the future is to make the multicolor LCM volume navigation real time. The parallel processing capabilities of the graphics work-stations can be used to make this remote goal easier.

One could extend the TINVIZ to handle more than three LCM volumes. Here, the color assignment to each channel is a very crucial task. While combining the channels one should be careful that the combined colors should not overlap the color tables used for different channels.

More accurate classification techniques than just the transparency mappings can be developed. Accurate volumetric measurements can serve biologists a great deal. Volumetric data compression can also help reducing the impact of very large multicolor LCM data sets.

6 Acknowledgements

We wish to acknowledge the following that helped make this work possible. OVPR (Office of Vice-Provost for Research, ASU), National Science Foundation's KDI grant (IIS-998016), NIH grant HD32621, Arizona Alzheimers Disease Research Center, Chuck Kazilek, the W. M. Keck Bioimaging Facility, CAGD and PRISM (Partnership for Research In Stereo Modeling) labs at ASU. We also wish to thank Drs. Robert Robertson and Dennis McDaniel for data in Figure 6.

7 References

[1] T. Brelje, M. Wessendorf, and R. Sorenson, “Multicolor Laser Scanning Confocal Immunofluorescence Microscopy” Practical Application and Limitations”, Cell Biological Applications of Confocal Microscopy, Methods in Cell Biology, Edited by Matsumoto, B. Vol. 38, 1993, pp. 98-177.

[2] G. T. Herman and H. K. Liu. Three Dimensional Display of Human Organs from Computer Tomograms. Computer Graphics and Image Processing, January 1979, pp. 1-21.

[3] W. Lorenson and H. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. Computer Graphics (Proceedings of SIGGRAPH `87), July 1987, pp. 163-169.

[4] M. W. Vannier, J. L. Marsh and J. O. Warren. Three Dimensional Computer Graphics for Craniofacial Surgical Planning and Evaluation. Computer Graphics, July 1983, pp. 263-273.

[5] M. Levoy, "Display of Surfaces from Volume Data", IEEE Computer Graphics and Applications, Vol. 8, No. 3, May 1988, pp. 29-37.

[6] L. Westover. Footprint Evaluation for Volume Rendering. Computer Graphics, August 1990, pp. 367-376.

[7] G. Nielson, "The Volume Rendering Equations", tutorial for inclusion in the notes for CSE 573, Spring 1998

[8] R. B. Haber and D. A. McNabb. Visualisation Idioms: A Conceptual Model for Scientific Visualization Systems. Visualisation in Scientific Computing, IEEE Press, Los Alamitos, CA, 1990.

[9] G. Sakas, M. Grimm and A. Savopoulos. "Optimized Maximum Intensity Projection (MIP)".

P. Hanrahan, W. Purgathofer (Eds.) "Rendering Techniques '95", Springer Verlag Wien/New York, pp. 51-63, 1995

[10] G. Sakas, G. Vicker, J. Plath. "Case Study: Visualization of Laser Confocal Microscopy Datasets". Proceedings IEEE Visualization '96, pp. 375-380, San Francisco, October 1996

[11] A. Jain. "Fundamentals of Digital Image Processing", 1989. Prentice Hall Information and System Sciences Series.

[12] G. Farin. “Curves and Surfaces for Computer Aided Geometric Design A Practical Guide”, Fourth Edition, Academic Press, 1996.