Fifth Irish Workshop on Computer Graphics (2004)
Eurographics Irish Chapter (Editors)

3D Visualisation of Confocal Fluorescence Microscopy Data

K. O'Conor[†], H. P. Voorheis[‡] and C. O'Sullivan[†]

[†] Image Synthesis Group, Department of Computer Science, Trinity College Dublin, Ireland
[‡] Cell Membrane Group, Department of Biochemistry, Trinity College Dublin, Ireland

Abstract

Confocal microscopes are able to non-invasively capture sub-micron details of fluorescent-labelled specimens at multiple depths, but there is a lack of applications capable of displaying this collected information in a meaningful and useful manner. We present a hardware-accelerated volume visualisation application developed for displaying and exploring confocal fluorescence microscopy data. Programmable graphics hardware is employed to improve visual quality and aid in highlighting significant properties of the volume, and an adapted marching cubes algorithm implemented to generate polygonal isosurfaces.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Virtual Reality

1. Introduction

The essential principle of confocal fluorescence microscopy relies upon the fact that there are two pinholes in the optical path, the first intercepts the light beam after leaving the light source and before striking the specimen and the second intercepts the light after leaving the specimen and before entering the detector. Then, at any fixed distance between pinholes and fixed position of lenses, only one plane normal to the light path will be in focus within the specimen, i.e. will be confocal with respect to both pinhole positions. In the commercial instruments available, the most common way to produce optical sections is to move the specimen in successive steps in a line between the two pinholes, termed the "z" direction, thus bringing into focus successive planes within the specimen. The thickness of the plane in focus is inversely related to the diameter of the pinholes and has a practical limit of about 0.2 of a micron. This optical arrangement effectively removes light from the planes within the specimen that are not in focus, commonly known as stray light and, consequently, sharpens the image at the focal plane recorded by the detector. Most of the image recording methods rely upon a roster scan of each of these optical z-planes using a laser light source and a photomultiplier detector coupled to a digitizer to produce a set of ordered datasets. Each dataset consists of an ordered array of data pairs that specify position and optical intensity at that position.

Various methods of presenting the data have been used in the past. These methods range from displaying each optical section as elements in a planar array of images arranged in step-order of z-direction, to creation of a pair of stereoscopic views by reconstructing the image from two separate angles through the three dimensional dataset. The angles are chosen to approximate those formed by a pair of human eyes viewing a specimen. Finally, the most common approach has been simply to superimpose all of the z-stack images in one single image. This last technique produces a sharper image than can be collected with ordinary epi-fluorescence microscopy because of the elimination of stray light in each plane, but has the disadvantage of loosing the z-directional information. New methods of presenting the data that reconstruct a partially transparent 3-dimensional image with perspective in the z-direction for ease of publication and easy viewing by the reader would be highly desirable.

In recent years research in the area of volume visualisation has advanced rapidly due to the increased availability of high speed, low-cost commodity graphics hardware. Volume data that could previously only be sufficiently explored by expensive workstation hardware can now be rendered at interactive rates on a common desktop PC. Although much of this work is aimed at visualising medical scans acquired by CT (Computerised Tomography) or MRI (Magnetic Resonance Imaging), many of the same methods can be applied to the display of confocal fluorescence microscopy data.

We will concentrate on three sections; the actual volume visualisation of collected data, implementing transfer functions to highlight individual surfaces and structures within

the volume, and reconstructing polygonal surfaces from the density data.

2. Volume Visualisation

Current methods for visualising confocal data such as stereoscopic cross-eyed images (where the user must cross their eyes in order to view the image in 3d - Figure 1) are difficult to use and do not lend themselves well to viewing or publication, nor allow for interactive investigation of the volume. By contrast, the stacks of image data collected by confocal microscopes can be readily used by texture mapping hardware to create an interactive and customisable rendering of the entire volume (Figure 2).

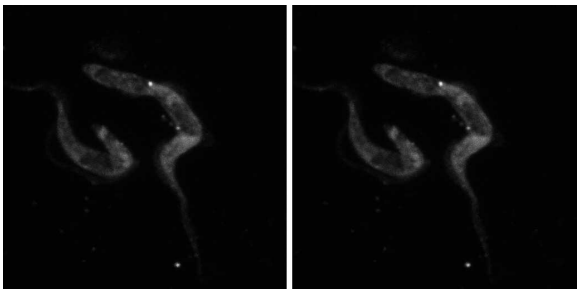

Figure 1: *Cross-eyed stereoscopic rendering of Trypanosoma brucei.*

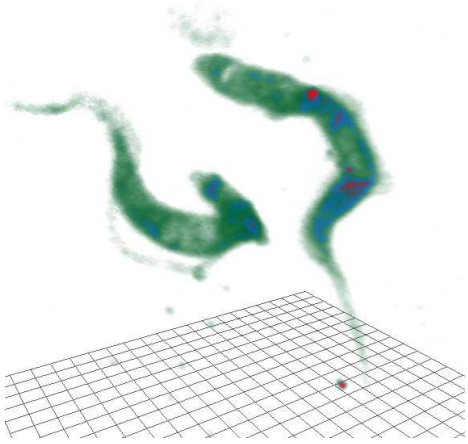

Figure 2: *Our visualisation of T. brucei.*

2.1. Textured slices

The basic method of visualising stacks of images such as these is by using 2D-textured object-aligned slices [KEHRS02]. The data collected by the microscope is a single 8 or 16-bit density value for each pixel and in basic direct volume rendering [EKE01] this value is mapped directly to a value such as colour or opacity. Proxy geometry is generated in the form of a quad for each texture slice, and these are arranged to give the illusion of a solid volume by blending each slice from back to front into the framebuffer.

However when the volume is rotated past a certain point, gaps between the slices become apparent (Figure 3) and the proxy geometry becomes evident. To avoid this 3 stacks must be generated, one for each principal axis. The original stack is used for the Z-axis, with interpolated stacks generated from this data used for the X and Y-axes. As the volume is rotated, the stack that is most orthogonal to the viewing direction is displayed, thus preserving the illusion of solidity. The main drawback of this method is that it consumes 3 times the amount of texture memory as the original stack, but this is unlikely to cause a major problem given the large amount of texture memory on modern graphics cards together with the increasing amount of bandwidth available for mapping AGP and PCI-Express texture memory. The alternative of using 3D textures would alleviate this problem and also allow trilinear filtering, but at the expense of frame rates [RE02] and trilinear filtering can be achieved anyway with 2D textures as outlined in the next section.

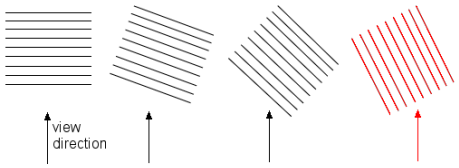

Figure 3: *Gaps would be visible between slices in the image on the right.*

2.2. Interpolated slices

Given the sub-micron scale of data generally collected by confocal microscopes, often the resolution limits the number of slices collected to a small amount. Displaying this data while preserving the scale and spacing of slices in the original specimen produces gaps and banding artefacts. While these problems can be remedied by inserting extra interpolated slices upon initial generation of the principal-axis stacks, this increases the amount of texture memory used unnecessarily by a factor of two or more, depending on the number of slices inserted.

Instead, slices can be inserted during run-time and interpolated without extra memory usage by employing multitexturing and programmable graphics hardware. We use NVidia's Register Combiners [NVI] to perform this interpolation. Register Combiners consist of a chain of 4-input general combiners (which have separate RGB and Alpha portions) where each combiner is set by the programmer to

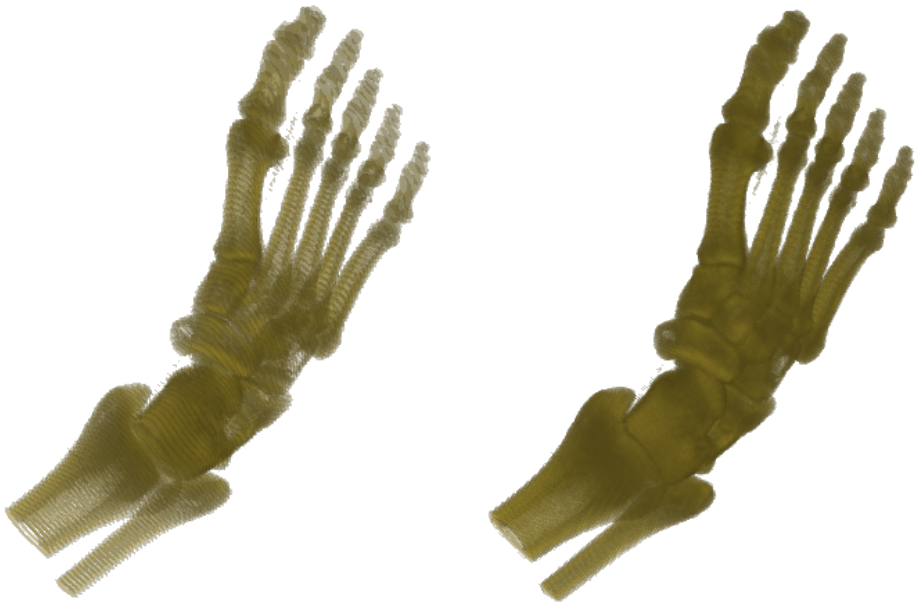

Figure 4: *One interpolated slice inserted.*

perform operations such as multiplication, addition and dot products on the data being passed through. After the general combiners, a final combiner can be used to perform interpolation on the output of the previous combiner stages. Possible inputs to any stage include texture lookups, primary/secondary colours and application-set constants. It is therefore possible to use the final combiner to interpolate between two textures slices by assigning the start and end textures to inputs B and C respectively, and an interpolation factor to input A signifying the distance of the inserted slice between the two existing slices (Figure 5).

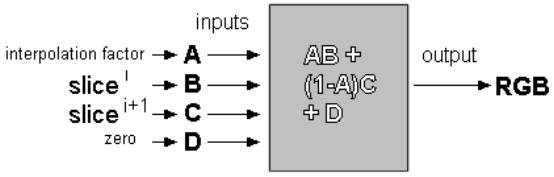

Figure 5: *Final Register Combiner interpolation setup.*

Bilinear image filtering can be performed on 2D textures by graphics hardware. By combining this with the linearly interpolated extra slices, the end effect is trilinear interpolation with similar results to that performed with 3D textures, but at improved frame rates. The number of extra slices inserted can be altered during run-time, and in order to preserve interactive frame rates these interpolated slices are only displayed when user input is not being received, so a large number of extra slices can be introduced to achieve dramatically increased image quality without hindrance to the user. We can see in Figure 4 the significant increase in quality that even one extra slice can give.

3. Transfer Functions

In order to convert the density values of the initial microscopy data into meaningful optical properties such as colour and opacity, a transfer function is needed. Transfer functions are generally implemented as one-dimensional colour lookup tables, mapping density directly to an RGBA value.

3.1. Classification

The transfer function can be baked directly into the texture data upon initial generation of the stacks, known as pre-classification. However this method is inefficient, as the entire volume needs to be updated whenever the transfer function is altered. In addition, pre-classification leads to artefacts in interpolation. An interpolated pre-classified colour might not be the same as would result from evaluation of an interpolated density value by the transfer function. Post-classification is used instead, where the data is stored in the textures and only converted into optical properties upon rendering, and after interpolation.

Post-classification can be accomplished in one of two ways. The first is to use OpenGL colour tables, where an 8-bit texel intensity value is used to look up a one-dimensional

colour table. The alternative is to use dependent texture reads, where a portion of the fragment colour resulting from one texture lookup are used as texture coordinates for a second texture. While somewhat slower, dependent texturing is more flexible than GL colour tables, and can be used for implementing multi-dimensional transfer functions [KKH01].

3.2. Dependent texturing with texture shaders

NVidia's Texture Shaders [NVI] extend OpenGL's standard texture addressing operations, allowing for an additional variety of operation such as offsetting, dependent texturing and dot product operations. There are two different types of dependent texture shaders, Alpha-Red and Blue-Green. These determine which colour channels of the first texture unit are used as texture coordinates for the second unit's addressing operation (Figure 6).

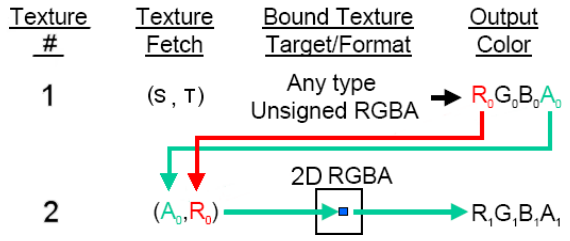

Figure 6: *Alpha-Red dependent texturing.*

We implement the transfer function using an Alpha-Red texture shader. When the initial textures are being uploaded, their internal format is set to GL_ALPHA8, so each density value is stored as an 8-bit alpha value. This is bound to the first texture unit, and a one-dimensional RGBA texture representing the transfer function is bound to the second unit. When texture shaders are enabled, the alpha value of the slice texture is converted into the corresponding colour and opacity according to the transfer function texture. This texture can be edited while the application is running and re-uploaded to alter the visualised colours in real-time. An example of the transfer function used for Figure 2 can be seen in Figure 7, upon which is superimposed a logarithmic histogram of density values present in the volume. While opacity information can be included in the transfer function texture and applied to the resulting visualisation, we have found that given the inherent noisy nature of confocal microscopy data, often better results can be obtained by using purely opaque colours.

4. Surface Reconstruction

The reason the volume data collected can be visualised by 2D texture slices is that it is arranged as a uniform 3D grid of values. This uniformity can be exploited to extract isosurfaces - continuous surfaces defined by points of equal value.

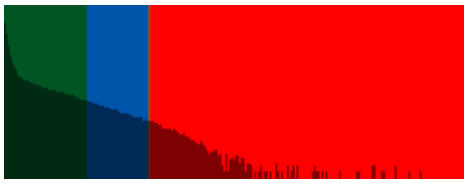

Figure 7: *Transfer function with superimposed logarithmic histogram.*

4.1. Marching Cubes

The marching cube algorithm [LC87] proceeds through the volume, processing one cube at a time. Each cube is composed of vertices corresponding to 8 pixels of the volume data, four each from two adjacent slices. The algorithm marches through every cube in the volume, determining if the specified isosurface intersects the current cube or not. With the two states (intersecting or not intersecting) and 8 vertices, there are a total of 256 possible ways a surface can intersect a cube. Eliminating those duplicated by symmetry and other similarities reduces the number of possible cases to a more manageable 14 (Figure 8). These are stored in a lookup table and triangles generated from this. However, in some of these cases there is the potential for ambiguity when there are more than one way to triangulate a given voxel, and this can lead to holes in the generated surface.

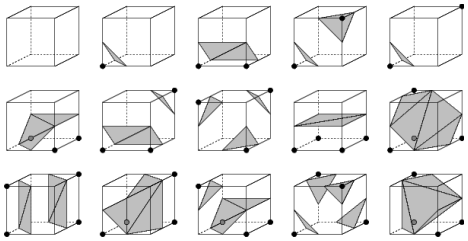

Figure 8: *The 14 possible cases of an isosurface intersecting a cube with Marching Cubes.*

4.2. Marching tetrahedra

To avoid these problems, an adaptation of the marching cubes algorithm changes the fundamental shape to a tetrahedron [GH95]. Each cube can be decomposed into five tetrahedra, and the number of possible triangulation cases is also reduced to five (Figure 9). Although this produces up to twice the number of triangles than marching cubes, ambiguous cases are avoided and a smoother mesh is produced.

See Figure 10 for an example of the marching tetrahedra algorithm applied to Trypanosoma brucei, along with the original volume the surface was extracted from.

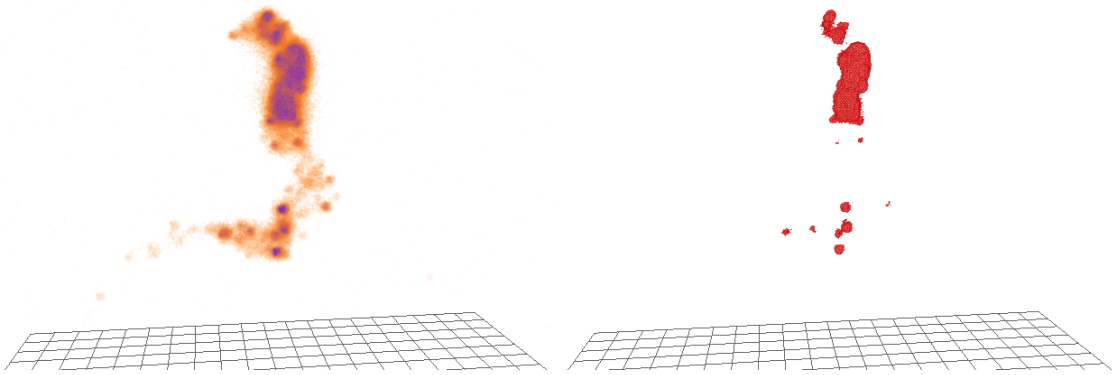

Figure 10: *Full volume (left) and wireframe of extracted isosurface (right).*

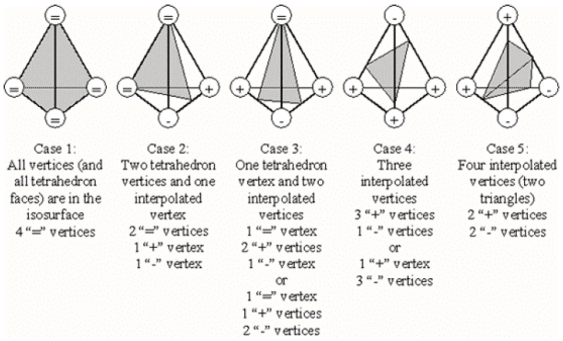

Figure 9: *The five possible cases using tetrahedra.*

5. Conclusions and Future Work

We have presented here an overview of a volume visualisation system for generating publishable images of confocal fluorescence microscopy data. It uses programmable graphics hardware in the form of texture shaders and register combiners to achieve a performance boost and improve image quality, as well as implementing an adaptation of the marching cubes algorithm in order to generate polygonal surfaces of equal isovalues.

Possible future extensions include:
Image filtering: As mentioned above, data collected by a confocal microscope tends to have a relatively low signal to noise ratio. This leads to incomplete isosurfaces and a lot of unnecessary data being visualised that could be removed to improve image fidelity. Image filtering such as a median filter [RPFC01] could be used to 'clean up' the data before visualisation.

Transfer function selection: The choice of transfer function makes a large impact on the resulting image, and is of primary importance when deciding what parts of the volume are highlighted for the viewer. Therefore developing a quick and intuitive way of specifying the transfer function interactively [Kin02] is a major concern.

Isosurface simplification: Given that much of the time the noisy data collected by confocal microscopy produces overly complex polygonal isosurfaces when the marching tetrahedron method is applied, implementing a method for reducing the polygon count [WHD02] would greatly increase frame rates as well as producing a cleaner image.

6. Acknowledgements

We would like to acknowledge the financial support of the Higher Education Authority of Ireland for making this project possible.

References

[EKE01] ENGEL K., KRAUS M., ERTL T.: High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *Proc. of Eurographics/SIGGRAPH Workshop on Graphics Hardware* (August 2001), vol. 2. 2

[GH95] GUEZIEC A., HUMMEL. R.: Exploiting triangulated surface extraction using tetrahedral decomposition. *IEEE Trans. Vis. Comp. Graph. 1*, 4 (1995), 328–342. 4

[KEHRS02] KNISS J. M., ENGEL K., HADWIGER M., REZK-SALAMA C.: High-quality volume graphics on consumer pc hardware. *Siggraph Course notes 42* (2002). 2

[Kin02] KINDLMANN G.: Transfer functions in direct volume rendering: Design, interface, interaction. *Siggraph Course Notes* (2002). 5

[KKH01] KNISS J. M., KINDLMANN G., HANSEN C.: Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. *Proceedings of IEEE Visualization* (2001), 255–262. 4

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes a high resolution 3d surface construction algorithm. *Computer Graphics 21*, 4 (July 1987), 163–169. 4

[NVI] NVIDIA: Nvidia opengl specifications. http://www.nvidia.com/Developer. 2, 4

[RE02] ROETTGER S., ERTL T.: A two-step approach for interactive pre-integrated volume rendering of unstructured grids. *In Symposium on Volume Visualization and Graphics* (2002), 23–28. 2

[RPFC01] RAZDAN A., PATEL K., FARIN G. E., CAPCO D. G.: Volume visualization of multicolor laser confocal microscope data. *Computers and Graphics 25*, 3 (2001), 371–382. 5

[WHD02] WOOD Z., HOPPE H., DESBRUN M.: *Isosurface Topology Simplification*. Tech. Rep. TR-2002-28, Microsoft Research, 2002. 5