Implementing Direct Volume Visualisation with Spatial Classification

Daniel Mueller
School of Electrical and Electronic Systems Engineering, QUT
Brisbane, QLD, Australia
d.mueller@qut.edu.au

Anthony Maeder
e-Health Research Centre,
CSIRO ICT Centre
Brisbane, QLD, Australia
anthony.maeder@csiro.au

Peter O'Shea
School of Electrical and Electronic Systems Engineering, QUT
Brisbane, QLD, Australia
pj.oshea@qut.edu.au

Abstract

Direct volume rendering (DVR) provides medical users with insight into datasets by creating a 3-D representation from a set of 2-D image slices (such as CT or MRI). This visualisation technique has been used to aid various medical diagnostic and therapy planning tasks. Volume rendering has recently become faster and more affordable with the advent of 3-D texture-mapping on commodity graphics hardware. Current implementations of the DVR algorithm on such hardware allow users to classify sample points (known as "voxels") using 2-D transfer functions (functions based on sample intensity and sample intensity gradient magnitude). However, such 2-D transfer functions inherently ignore spatial information. We present a novel modification to 3-D texture-based volume rendering allowing users to classify fuzzy-segmented, overlapping regions with independent 2-D transfer functions. This modification improves direct volume rendering by allowing for more sophisticated classification using spatial information.

INTRODUCTION

Broadly speaking, visualisation is an iterative process in which the user undertakes the tasks of exploration, analysis and presentation [1]. Human pattern recognition processes, relying on visual sensory input from such visualisations, provide a means of understanding complex anatomical and physiological situations. Direct volume rendering is a visualisation technique that is useful in a variety of medical situations including virtual endoscopy [2], 3-D ultrasound [3], and surgical planning [4, 5]. Consequently we seek ways to improve the visualisation of medical datasets using direct volume rendering.

Volume rendering begins by sampling a continuous object of interest (such as a human appendage) and forming a discrete spatial model. The medical domain has various imaging modalities capable of performing such sampling including X-ray computed tomography (CT) and magnetic resonance imaging (MRI). Each discrete sample in the 3-D model is referred to as a "voxel" (volume element). Classical medical diagnosis and therapy planning is undertaken by viewing individual 2-D slices of the sampled data. Volume rendering allows for further insight by converting the data model into interactive 2-D photorealistic renditions. These renditions are formed by modelling each voxel as a semi-transparent light emitting particle, and observing the virtual light projected onto an image viewing plane (refer to Figure 1) [6]. Before projecting each voxel contribution, a user specified classification function is applied to enhance different structures of interest. This function (commonly referred to as a "transfer function") assigns colour and opacity to each voxel, dependent of various attributes (for example a 2-D transfer function uses sample intensity and sample intensity gradient magnitude).

Traditionally, one of the major hurdles associated with volume rendering was the high computational expense of the rendering algorithms [7]. The introduction of 3-D texture-mapping capabilities to commodity graphics hardware has allowed for a faster and more affordable implementation. The current implementation uploads a dataset to the graphics processing unit (GPU) which in turn performs the millions of trilinear-interpolations in a highly parallel nature. This method allows for the interactive exploration of visualisation parameters including rotation, translation, zoom, and classification.

The GPU implementation of direct volume rendering currently only allows for the application of global classification functions – users cannot emphasise important spatial features using these global functions. We present a novel approach allowing the user to specify spatially independent 2-D transfer functions. Prior to visualisation the user fuzzy-segments a number of regions, each of which is subsequently assigned an independent classification function. This method allows users to spatially classify and visualise a volume dataset.

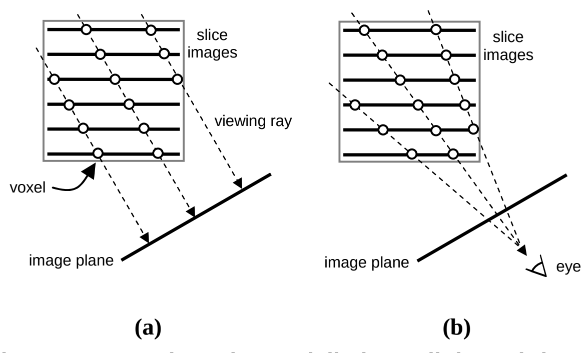


Figure 1. A voxel can be modelled as a light emitting particle and projected onto an image plane.
(a) Parallel Projection (b) Perspective Projection.

RELATED WORK

Traditional transfer functions are applied on the global level, ignoring the spatial domain for sake of ease. Recently "dual-domain" interaction was introduced whereby the user probes the spatial domain to aid with the construction of a global 2-D transfer function [8]. However, this approach still applies the transfer function in a global fashion to all voxels. A different approach tags each voxel with an identifier pointing to one of *n* transfer functions associated with different regions [6]. Unfortunately this approach only caters for hard-segmented, non-overlapping regions and is best suited to pre-classification implementation.

We present a modification to 3-D texture-based volume rendering overcoming the disadvantages discussed above. Our proposed approach allows for spatial 2-D classification using hard- and/or fuzzy-segmented, overlapping regions.

3-D TEXTURE-BASED VOLUME RENDERING

3-D texture-based volume rendering is primarily executed on the graphics hardware. The data is uploaded to hardware memory as a set of 2-D slice images via an API (application programmers interface) such as OpenGL or Direct3D. Through the API, a user program outputs view-aligned polygons which act as an equidistant, rectilinear grid capable of tri-linearly interpolating the uploaded data at any viewing angle. The sampling rate determines the distance between the grid intersection points of this "proxy-geometry" (see Figure 2). The interpolated view-aligned image slices (of which elements are referred to as "fragments") are finally composited into a single 2-D rendition using the operation defined in Equation (1) [6]. For a typical dataset of 256^3 voxels the rendering algorithm must perform millions of tri-linear interpolations. By taking advantage of the parallel architecture of modern GPUs, it is possible to execute the rendering algorithm at real-time framerates (≈25 fps).

$$
\begin{aligned}
c_{final} &= \sum_{i=0}^{n-1} c_i \times \prod_{j=0}^{i-1} 1-\alpha_j \\
&= c_0 + c_1(1-\alpha_0) + c_2(1-\alpha_0)(1-\alpha_1) + \ldots \\
&= c_0 \; over \; c_1 \; over \; c_2 \; over \ldots over \; c_{n-1}
\end{aligned}
\qquad \textbf{(1)}
$$

where :

α_i is the opacity of sample i

C_i is the RGB colour of sample i

$C_i\alpha_i = c_i$ is the pre - multiplied colour and opacity of sample i

c_{final} is the final colour composed from all samples

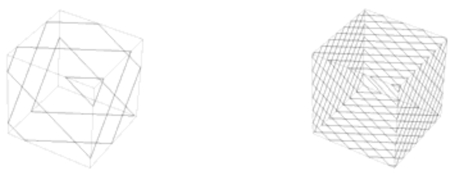


Figure 2. A set of parallel view-aligned polygons act as proxy-geometry with (a) low sampling rate, and (b) higher sampling rate.

MULTIDIMENSIONAL CLASSIFICATION

Classification transfer functions allow data to be made visible and hence play a pivotal role in volume rendering. Basically a transfer function acts as a filter to colour important information and suppress the visibility of unwanted noise. Transfer function specification has emerged as an important research topic [8, 9].

Multidimensional transfer functions (in particular 2-D functions) have become a popular choice for volume classification. Medical datasets typically contain information pertaining to complex interactions between boundaries of different materials. A 1-D transfer function is unable to isolate a voxel belonging to multiple boundaries [8]. A 2-D transfer function on the other hand, specifies the colour and opacity of voxels based on sample intensity and sample intensity gradient magnitude, allowing for the isolation of more than one boundary. A global 2-D transfer function can be considered as a lookup table (LUT) which returns an RGB colour (c) and opacity (α) for the given lookup values f and f' as defined in Equation (2) below.

$$\forall \text{ voxels } v_{ijk} : \{c, \alpha\} = T(f, f')$$

where :

v_{ijk} is voxel at location (x_i, y_j, z_k)

c is the returned RGB colour

α is the returned opacity **(2)**

T denotes the "transfer function"

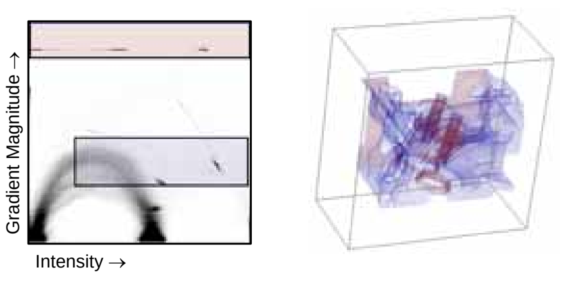
f is the data intensity of voxel v_{ijk}

f' is the gradient magnitude of f defined as

$$f' = |\nabla f| = \sqrt{\left(\frac{\partial}{\partial x} f\right)^2 + \left(\frac{\partial}{\partial y} f\right)^2 + \left(\frac{\partial}{\partial y} f\right)^2}$$

There are two types of classification: (a) pre-interpolation classification (also referred to as "pre-classification") in which the voxel is assigned colour and opacity *before* the interpolation operation, and (b) post-interpolation classification (also referred to as "post-classification") in which the voxel is assigned colour and opacity *after* the interpolation operation. 3-D texture-based volume rendering can be implemented with both pre- and post-classification, however it has been shown that post-classification produces superior results [6].

Implementing post-classification volume rendering using 3-D textures and graphics hardware requires the use of a fragment shader program (which can be considered as a kernel function applied to all interpolated voxels). These "per-fragment" operations are performed by the GPU as the final step of the hardware pipeline. The data is uploaded to the hardware as a 3-D texture. A 3-D texture is a set of image slices with each pixel consisting of four channels (RGBA = Red, Green, Blue, Alpha). While labelled RGBA, these channels are not restricted to colour information alone. In our case, the dataset is uploaded to the graphics hardware using the Alpha channel for sample intensity (f) and the Red channel for sample intensity gradient magnitude (f'). Along with this, a 2-D texture is created to serve as the actual transfer function lookup table. The x-axis corresponds to sample intensity and the y-axis to sample intensity gradient magnitude (refer to Figure 3).



(a) (b)

Figure 3. (a) A 2-D transfer function (excluding the underlaid histogram) is uploaded to the GPU as a LUT texture. (b) An example rendition of the engine dataset using the transfer function in (a).

During the per-fragment operation phase of the graphics pipeline, the fragment shader interpolates the data and gradient information. This interpolated information is in turn used as lookup values for the transfer function (uploaded as a 2-D texture). The returned value from the lookup table is set as the fragment RGBA colour. The fragments are then composited together along viewing rays using Equation (1), as previously discussed. Listing 1 shows the fragment shader program for such an operation.

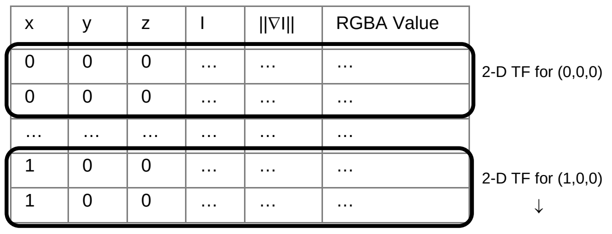
```
//Texture samplers
uniform sampler3D sampler_data;
uniform sampler2D sampler_tf;

//Texture coordinates from vertex shader
varying vec3 data_coord;

//-------------------------------------------------
//Function: main
//Description: Fragment operation for 2-D
//             classification function
//-------------------------------------------------
void main()
{
    //Use coord to interpolate data & gradient
    vec4 data = texture3D(sampler_data, vec3(data_coord));

    //Setup data & gradient as TF lookup values
    //Data = Alpha channel
    //Gradient = Red channel
    vec2 tf_coords = vec2(data.a, data.r);

    //Look up 2-D transfer function LUT
    vec4 tf_data = texture2D(sampler_tf, tf_coords);

    //Set fragment colour and opacity from lookup value
    gl_FragColor = tf_data;
}
//-------------------------------------------------
```

Listing 1. The fragment shader program for a 2-D transfer function (OpenGL Shading Language).

SPATIAL CLASSIFICATION

From the discussion so far, it can be seen how to implement volume rendering with 2-D post-classification using accelerated graphics hardware. However, this implementation does not allow for the classification of voxels at specific spatial locations. Such a capability is desirable for more sophisticated visualisation.

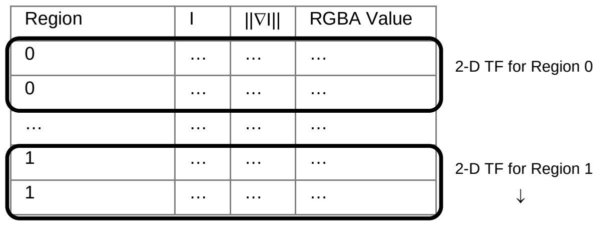
At first glance a simple solution might be to extend our 2-D transfer function to include (x, y, z) components, creating a 5-D classification function. While this may provide the desired functionality, this solution can not currently be implemented on graphics hardware. For a typical dataset of 256^3 voxels and 8-bit lookup values, a 5-D transfer function LUT would require $256^5 \times 8 \approx 8.796 \times 10^{12} \, bits \approx 81{,}262 \; GB$ of memory. This far exceeds the 256 MB of memory on current graphics cards. Fortunately such an approach is not required because much of a 5-D transfer function LUT would contain redundant spatial information. Our solution proposes to group (x, y, z) entires in a 5-D lookup table into regions. A 5-D transfer function allows the user to specify an independent 2-D transfer function for each possible (x, y, z) coordinate. Using our region-based approach, users are only required to assign an independent 2-D transfer function to each segmented region. The memory requirement for such a system is $256^2 \times n \times 8 \; bits$, where n is the number of regions. This grouping not only significantly reduces the memory requirements of the algorithm, but it is also more intuitive to the user.



x	y	z	I	‖∇I‖	RGBA Value
0	0	0	…	…	…
0	0	0	…	…	…
…	…	…	…	…	…
1	0	0	…	…	…
1	0	0	…	…	…

2-D TF for (0,0,0)
2-D TF for (1,0,0)
↓

(a)



Region	I	‖∇I‖	RGBA Value
0	…	…	…
0	…	…	…
…	…	…	…
1	…	…	…
1	…	…	…

2-D TF for Region 0
2-D TF for Region 1
↓

(b)

Figure 4. (a) A 5-D transfer function has a unique 2-D LUT for each (x,y,z) coordinate which is superfluous (b) A region-based TF has a unique 2-D LUT *for each region*, which significantly reduces the memory requirements while not severly effecting functionality.

METHOD

The proposed approach was realized on accelerated graphics hardware using a number of additional textures. An additional four channel (RGBA) 3-D texture was used to support up to four 8-bit greyscale region masks (one per channel). These region masks allow users to create fuzzy regions (white indicates that the associated voxel does not belong to the region, black indicates that the voxel does belong to the region, and the grey continuum in between indicates varying degrees of membership). Figure 6 and Figure 7 show some example masks. Each region must also add an extra 2-D texture to store the independent 2-D transfer function. Additional proxy-geometry must also be output to interpolate the new 3-D region texture. Our approach proposes the novel idea of interweaving slices of both data and regions together. Figure 5 depicts a comparison of our proposed approach and the typical approach.

A fragment shader for the proposed algorithm uses the discussed texture structure to facilitate the spatial classification. Both the data/gradient and region 3-D textures are interpolated by the shader program. Following this, each region-based transfer function is sampled using the interpolated data/gradient value. This determines the colour and opacity for the current fragment for each region. Next, the colour and opacities for each region are weighted by the associated region mask and combined into a final colour for the current fragment. This operation is detailed for opacities in Equation (3) (colour components are treated in an identical fashion). Finally, the view-aligned, region-weighted slice images are composited in the normal manner described by Equation (1).

$$\begin{aligned}\alpha_{out} &= \sum_{i=0}^{n-1}(\omega_i\alpha_i)\times\prod_{j=0}^{i-1}(1-\omega_j\alpha_j)\\ &= (w_0\alpha_0)+(w_1\alpha_1)(1-w_0\alpha_0)+(w_2\alpha_2)(1-w_1\alpha_1)(1-w_0\alpha_0)+\dots\end{aligned} \qquad (3)$$

where:

ω_i is the region mask weight from the region texture

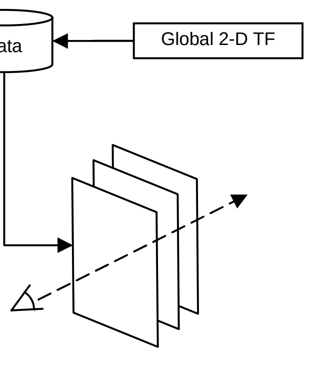
α_i is the opacity value from the transfer function LUT

RESULTS AND DISCUSSION

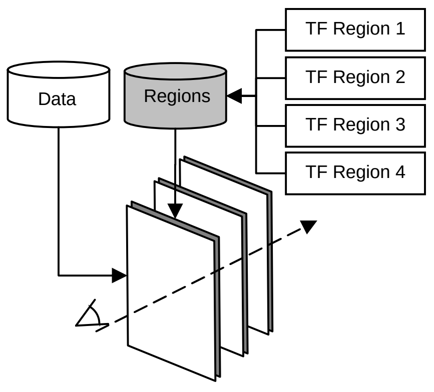
For validation purposes the proposed algorithm was implemented on an ATI Radeon 9800 Pro GPU and Intel Pentium 4 2.8GHz, 1GB RAM PC using OpenGL. This proof-of-concept implementation was tested using two common volume rendering datasets with simple box and spherical regions. Figure 6 and Figure 7 depict renditions using the traditional and proposed algorithms with two regions.

Visual inspection confirms that classification has only been performed for the desired region(s). The major strength of the proposed algorithm is the ability to attach independent 2-D transfer functions to different regions of interest. These regions can be fuzzy-segmented catering for uncertainty involved with segmentation. Furthermore, regions may overlap allowing for complex classification of volume data.

There is however, a trade-off between performance and functionality, as reflected in Table 1. The proposed algorithm is executed three times slower than the traditional algorithm. Unfortunately this is expected due to the additional texture interpolations required for supporting spatial classification.



(a)



(b)

Figure 5. (a) The typical texture layout for 3-D texture based volume rendering uses one 2-D transfer function for the data image slices (b) Our proposed approach interweaves the data and region 3-D textures, assigning a unique 2-D transfer function to each channel of the region texture.

Dataset	Algorithm	Volume Size	Image Size	Framerate
Engine	Traditional	256x256x128	256x256	25 fps
Engine	Spatial	256x256x128	256x256	8 fps
Foot	Traditional	256x256x256	256x256	9 fps
Foot	Spatial	256x256x256	256x256	3 fps

Table 1. A comparison of the framerates between the traditional and proposed algorithm reveals that the traditional is approximately 3 times faster. (Framerates were measured using a sampling rate of 1.5.)

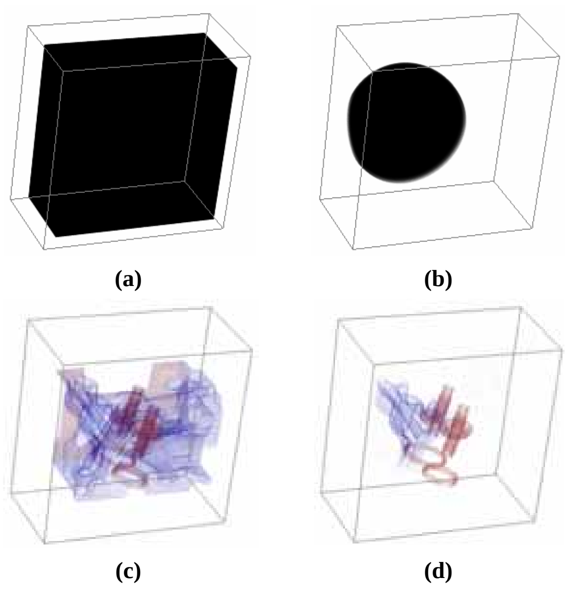


Figure 6. Results using the engine dataset [1].

(a) Region 1: hard-segmented box region

(b) Region 2: fuzzy-segmented spherical region

(c) Rendition *without* spatial classification

(d) Rendition *with* spatial classification (using region 1 & 2)

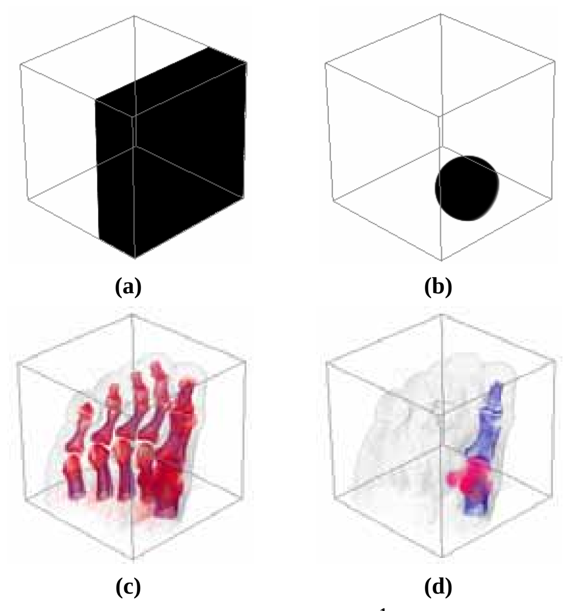


Figure 7. Results using the foot dataset [1].

(a) Region 1: hard-segmented box region

(b) Region 2: fuzzy-segmented spherical region

(c) Rendition *without* spatial classification

(d) Rendition *with* spatial classification (using region 1 & 2)

[1] Available from http://www.volvis.org/

CONCLUSIONS AND FUTURE WORK

We have presented a novel modification to 3-D texture-based volume rendering capable of performing spatial classification. Fuzzy-segmented, potentially overlapping regions can be assigned independent 2-D transfer functions. This approach allows for more sophisticated visualisation and can achieve interactive framerates. However, real-time framerates can not be achieved as in traditional GPU implementations due to the increased overheads.

Future work will endeavour to apply the proposed algorithm to clinical data and demonstrate the improved capabilities in diagnosis and therapy planning. To facilitate this step, more complex segmentation algorithms (such as region growing or fuzzy C-means) must be integrated into the framework.

REFERENCES

[1] P. Keller and M. Keller, *Visual cues: practical data visualization*. Los Alamitos, CA: IEEE Computer Society Press, 1993.

[2] J.-W. Hwang, J.-M. Lee, I.-Y. Kim, I.-H. Song, Y.-H. Lee, and S. Kim, "A PC-based high-quality and interactive virtual endoscopy navigating system using 3D texture based volume rendering," *Computer Methods and Programs in Biomedicine*, vol. 71, pp. 77-84, 2003.

[3] C. Kim, J. H. Oh, and H. Park, "Efficient volume visualization of 3D ultrasound images," presented at SPIE Medical Imaging: Image Display, 1999.

[4] R. A. Robb, "Three-dimensional visualization in medicine and biology," in *Handbook of Medical Imaging : Processing and Analysis*, I. N. Bankman, Ed. San Diego: Academic Press, pp. 685-712, 2000.

[5] F. V. Higuera, N. Sauber, B. Tomandl, C. Nimsky, G. Greiner, and P. Hastreiter, "Automatic adjustment of bidimensional transfer functions for direct volume visualization of intracranial aneurysms," presented at SPIE Medical Imaging: Visualization, Image-guided Procedures, and Display, San Diego, 2004.

[6] C. Rezk-Salama, "Volume rendering techniques for general purpose graphics hardware," PhD dissertation, Department of Computer Science 9 (Computer Graphics), University Erlangen-Nuremberg, 2001.

[7] P. G. Lacroute, "Fast volume rendering using a shear-warp factorization of the viewing transformation," PhD dissertation, Department of Electrical Engineering and Computer Science, Stanford University, 1995.

[8] J. Kniss, G. Kindlmann, and C. Hansen, "Multidimensional transfer functions for interactive volume rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, pp. 270-285, 2002.

[9] H. Pfister, B. Lorensen, C. Bajaj, G. Kindlmann, W. Schroeder, L. S. Avila, K. M. Raghu, R. Machiraju, and J. Lee, "The transfer function bake-off," *IEEE Computer Graphics and Applications*, vol. 21, pp. 16-22, 2001.