TEXTURE-ENHANCED DIRECT VOLUME RENDERING

Felix Manke and Burkhard Wünsche
Graphics Group, Department of Computer Science, University of Auckland, Private Bag 92019, Auckland, New Zealand

Keywords: Direct-volume rendering, Transfer Functions, Multi-field data, Texture synthesis, Texture morphing.

Abstract: Direct volume rendering (DVR) is a flexible technique for visualizing and exploring scientific and biomedical volumetric data sets. Transfer functions associate field values with colors and opacities, however, for complex data are often not sufficient for encoding all relevant information. We introduce a novel visualization technique termed texture-enhanced DVR to visualize supplementary data such as material properties and additional data fields. Smooth transitions in the underlying data are represented by coherently morphing textures within user defined regions of interest. The framework seamlessly integrates into the conventional DVR process, can be executed on the GPU, is extremely powerful and flexible, and enables entirely novel visualizations.

1 INTRODUCTION

Volumetric data is common in science, engineering and biomedicine. Visualizations help to gain insight and understanding of the data. A powerful and popular visualization technique is *Direct Volume Rendering* (DVR), which does not require intermediate representation and can display an entire 3D data set in a 2D image. This is achieved by associating data values with opacities and colors via *transfer functions*.

Traditionally DVR has only been used for scalar data. While extensions for higher dimensional data exist, the fact that only color and opacities can be used to represent field values presents a natural limitation. In this paper we present a novel concept to enrich DVR visualizations with textures, which are a separate visual attribute independent of color and opacity (Landy and Graham, 2004). The technique facilitates the visualization of higher-dimensional and multi-field data by encoding additional field values by texture attributes. Additionally, the actual appearances of different materials within the data set can be mimicked resulting in more realistic and intuitive visualizations. Textures in regions with overlapping transfer functions are morphed to create a smooth transition between different textured materials. Our mathematical framework represents a natural extension of the traditional DVR process and is consistent with existing opacity and color transfer functions.

Section 2 surveys previous work on visualizing complex data with textures and DVR. Section 3 reviews concepts of traditional DVR. Section 4 formally defines a mathematical framework for texture-enhanced DVR. Section 5 discusses the algorithmic structure of texture-enhanced DVR, gives implementation details, and discusses techniques to improve perception of 3D textures. Results and a discussion of our proposed concept follow in section 6. Section 7 concludes with a summary of our main contributions and important directions for future research.

2 RELATED WORK

Several modifications of traditional transfer functions have been suggested. Feature rich visualizations can be obtained by using multi-dimensional transfer functions and applying them to scalar or multivariate data (Kniss et al., 2002). Special manipulation widgets make the specification of transfer functions more intuitive and convenient.

Textures have been used previously in DVR to display higher-dimensional data. Vector fields can be represented with Line Integral Convolution (LIC) textures and interactive explored with DVR (Rezk-Salama et al., 1999). 3D Perception can be improved by emphasizing thin thread structures using *limb darkening* (Helgeland and Andreassen, 2004) or visibility-impeding halos which indicate depth discontinuities (Wenger et al., 2004).

Tensor fields can be represented by integrating streamlines along the principal eigenvector direction. The resulting textures can be blended with representa-

Manke F. and Wünsche B. (2009).
TEXTURE-ENHANCED DIRECT VOLUME RENDERING.
In *Proceedings of the Fourth International Conference on Computer Graphics Theory and Applications*, pages 185-190
DOI: 10.5220/0001772701850190
Copyright © SciTePress

tions for different tissue types by using derived tensor quantities as input for a classification function which encodes the probability that a field value corresponds to a certain tissue type (Wünsche and Lobb, 2004).

Patel et al. (Patel et al., 2007) illustrate volumetric seismic data by mapping 2D textures onto the planar faces of axis-aligned cutouts. For the rendering, "Layer texture transfer functions" are used to select textures, scaling factors, and opacities for the pre-classified layers. "Scalar texture transfer functions" associate different materials with textures. During rendering, the opacities are used to linearly blend the 2D textures. A separate conventional color-opacity transfer function maps data values to $RGB\alpha$ colors, which can be blended with the textures.

3 MATHEMATICAL MODEL FOR DVR

Traditional DVR is described by the emission-absorption model, where scalar values are interpreted as densities of a gaseous material which emits and absorbs light (Max, 1995). An image is created by accumulating the total light intensity for each pixel which in the simplest case is computed as

$$C = \int_0^\infty c(t) e^{-\int_0^t \kappa(u) du} dt \tag{1}$$

where t is the parameter of a viewing ray through a pixel, C is its color, $c(t)$ the color at the ray parameter t and the integral in the exponent is the total opacity of the ray segment $[0,t]$ which is computed by integrating densities (opacities) along the ray.

The DVR integral in equation 1 requires *transfer functions* which associate values of a volume with optical properties. In the traditional DVR model (Max, 1995) these are three color components (red, green, and blue) and an opacity component (α). Although Max assumes scalar volume data sets only, subsequent work uses higher dimensional input data such as vectors and tensors (Helgeland and Andreassen, 2004). Consequently we define a volume as a more general function $f: \mathbb{R}^n \mapsto \mathbb{R}^m$, with n independent variables, such as position and time, and m dependent variables, e.g., $m = 3$ for vector data.

In addition we introduce a data transformation operator $\triangleright$ that maps a function $f: \mathbb{R}^n \mapsto \mathbb{R}^m$ (input data set) to a function $g: \mathbb{R}^n \mapsto \mathbb{R}^k$ (derived data set):

$$\triangleright : f \mapsto g. \tag{2}$$

This operator reflects that many applications use derived quantities as input for the DVR process. For example, multi-dimensional transfer functions using scalar data and its gradient are useful for detecting material boundaries (Kniss et al., 2002).

The output of function g is a k-dimensional vector that serves as input for the transfer function. Using $\triangleright$, the color-opacity transfer function can be defined as:

$$\begin{gathered}\Theta : \mathbb{R}^k \mapsto \mathbb{R}^4 \\ \Theta\big(\triangleright(f)(P)\big) = \Theta\big(g(P)\big) = (C, \alpha).\end{gathered} \tag{3}$$

The transfer functions themselves are weighting functions for $RGB\alpha$ values and are usually either user-defined using simple manipulation widgets (Kniss et al., 2002), or are predefined, e.g., using typical tissue densities for CT data.

4 MATHEMATICAL FRAMEWORK FOR TEXTURE-ENHANCED DVR

In order to obtain consistency with the existing DVR model, *texture transfer functions* should enable the association of data values with different textures analogous to traditional transfer functions, i.e., by using weighting functions defined over the domain of the dependent variables.

The following difficulties exist: In contrast to colors and opacities, textures have spatial (and potentially temporal) properties and hence must also depend on the independent variables. Transfer functions which overlap within the domain of an independent variable, require a merging of textures. Whereas colors and opacity values can be (linearly) interpolated, the morphing of textures must maintain the characteristics of each individual texture used to represent the underlying data set, e.g., size, color and orientation of texture components. Since textures are also represented by color and opacities a mechanism must be found to combine them with the results of the color and opacity transfer functions.

The mathematical framework for texture-enhanced DVR is hence defined as follows:

4.1 Defining Texture Coordinates

Each texture object $\mathbf{T}(\Omega, \varsigma)$ has its own dimensionality p, domain Ω (where the texture is used in the visualization), and attribute function ς (usually $RGB\alpha$, but other values such as displacements are possible). Consequently, the texture coordinates Q have to be defined per texture. For volume rendering the texture coordinates depend on the spatial locations of the voxels. This requirement is consistent with existing texture-based visualization techniques such as LIC:

the texture generation only effects the attribute function ς, but not the definition of texture coordinates or weighting-curves used for rendering them.

In order to obtain texture coordinates for a texture $\mathbf{T}(\Omega,\varsigma)$, we define a function Γ that maps voxel positions $P \in \mathbb{R}^n$ to texture coordinates $Q \in \Omega \subseteq \mathbb{R}^p$:

$$\begin{aligned} \Gamma &: \mathbb{R}^n \mapsto \mathbb{R}^p \\ \Gamma &: P \mapsto \Gamma(P) = Q. \end{aligned} \tag{4}$$

Note that the equation makes no assumptions about the dimensionality of $\mathbf{T}$. In our examples $p = 3$ and Ω can be a subset of the input data set as explained in subsection 5.2. The definition of Γ allows spatially varying transformations, e.g. gradual scaling or rotation of textures and their features.

4.2 Texture Transfer Functions

Instead of returning colors and opacities, the new transfer function returns a vector of weights (one for each of the L textures) used for the classification:

$$\begin{aligned} \Theta_{tex} &: \mathbb{R}^k \mapsto \mathbb{R}^L \\ \Theta_{tex}\big(\rhd(f)(P)\big) &= \omega. \end{aligned} \tag{5}$$

Note that, in contrast to the conventional transfer function, Θ_{tex} does not map into $\mathbb{R}^4(RGB\alpha)$, but into an L-dimensional space ($\mathbb{R}^L$). This makes it possible to morph textures and define texture-dependent shading and transparency effects to improve perception.

4.3 Morphing of Textures

A morphing operator needs access to the texture transfer function Θ_{tex} in order to obtain weights for neighboring pixels. In addition morphing requires for all L textures the texture attributes at all texels, and not just their values at a given voxel. Hence, the operator needs L texture coordinates $Q_{1\cdots L}$ and the functions $\varsigma_{1\cdots L}$ as input. In order to further increase flexibility, it is better not to provide the pre-transformed texture coordinates, but instead the transformation functions $\Gamma_{1\cdots L}$ together with the current voxel position P. In this way, a morphing operator is able to access neighboring voxels and texels. Since the transformation functions $\Gamma_{1\cdots L}$ determine the texture domain the explicit specification of $\Omega_{1\cdots L}$ is not necessary.

Taken all these considerations into account, the morphing operator $\odot$ can be defined as follows:

$$\big((C,\alpha)_{tex},\ \Lambda\big) = \odot\big[P,\ \Gamma_{1\cdots L},\ \varsigma_{1\cdots L},\ f,\ \rhd,\ \Theta_{tex}\big]. \tag{6}$$

where Λ denotes additional channels such as displacement values. Note that for efficiency the morphing, even though defined for the entire volume, should be performed only for the subset of f which, according to the transfer functions will be visible and contribute to the final image (see subsection 5.2).

We have developed a fast exemplar-based texture synthesis and morphing algorithm. The technique provides an excellent trade-off between speed and quality, is highly flexible, allows the use of arbitrary channels, can be extended to arbitrary dimensions and is suitable for a GPU-implementation. Technical details are given in (Manke and Wünsche, 2009).

4.4 Color Combination Operator

The evaluation of the texture transfer function Θ_{tex} and the application of the morphing operator form a parallel path in the DVR process, which is independent of the evaluation of the color and opacity transfer function. In order to integrate texture-enhanced DVR into the existing process, the binary color combination operator $\circledast$ mixes the color-opacity pairs $(C,\alpha)_{conv}$ of the conventional transfer function and $(C,\alpha)_{tex}$ of the texture. Formally, $\circledast$ is defined as:

$$\begin{aligned} \circledast &: \ RGB\alpha \times RGB\alpha \mapsto RGB\alpha \\ (C,\alpha) &= (C,\alpha)_{conv} \ \circledast \ (C,\alpha)_{tex}. \end{aligned} \tag{7}$$

The color combination operator $\circledast$ has different modes implementing simple operations, such as replacement of the results of the color transfer function, and more complex operations combining colors and transparencies. This is consistent with the OpenGL texture environment modes `GL_REPLACE`, `GL_MODULATE` etc. for polygon rendering.

5 IMPLEMENTATION

The volume rendering integral in equation 1 can be efficiently solved on graphics hardware by discretizing the volume using object or view-aligned slices, computing colors and transparencies for each slice, and compositing them in back-to-front order:

```
$(C_i,\alpha_i)_{conv}$ <- evaluate $\Theta\big(\rhd(f)(P)\big)$
    // eval. of conv. transfer function
$\big((C,\alpha)_{tex},\ \Lambda\big)$ <- $\odot\big[P,\ \Gamma_{1\cdots L},\ \varsigma_{1\cdots L},\ f,\ \rhd,\ \Theta_{tex}\big]$
    // morphing of textures
$(C_i,\alpha_i)$ <- $(C_i,\alpha_i)_{conv} \circledast (C_i,\alpha_i)_{tex}$
    // combination of colors
$c'_i$ <- $(\alpha_i \cdot C_i) + (1-\alpha_i)\cdot c'_{i-1}$
    // volumetric compositing
```

The extensions can be directly integrated into a modular GPU-based DVR framework we presented previously (Manke and Wünsche, 2008). Sampling the texture transfer function generates weight maps which are used in the texture morphing algorithm.

$$\widetilde{\mathbf{W}}_j(P) \Leftrightarrow \Theta_{tex}\big(\triangleright(f)(P)\big) = \omega_j, \qquad 1 \le j \le L.$$

More implementation and technical details, and the source code are available at (Manke, 2008).

5.1 Perceptual Issues

One of the major difficulties in using 3D textures for visualization is perception of their spatial properties. While the human visual system is well adapted to perceive textures over surfaces and even uses them as depth and shape cues, it is very difficult to perceive partially transparent solid textures. The challenges are similar as for conventional color and opacity transfer functions where the naive approach of just accumulating colors and opacities results in a fuzzy image without visible structures. These problems are overcome by using shading functions which emphasize material boundaries, e.g., by using the gradient magnitude of the scalar input data set as pseudo surface normal (Levoy, 1988). Based on experimental and visual perception research we developed the following guidelines for using texture-enhanced DVR:

For data sets where only recognition of material types and properties is important, rather than their exact 3D structure, we use opacity transfer functions which make the region of interest nearly opaque and the remaining features nearly transparent. Interior structures can be shown by using cutting planes. The method results in images similar to Patel et al.'s approach (Patel et al., 2007), but offers smooth boundaries between materials using texture morphing and we can indicate important features or anatomical landmarks by partial transparent surfaces.

A better perception of the 3D structure of textures is obtained by using screen-door transparencies. Here, the alpha channel of the synthesized texture is used to model different opacities for the texture elements. For screen-door transparency, selected parts of the texture are defined to be fully opaque, whereas all other parts are defined to be (almost) fully transparent. The color combination operator then multiplies the opacities α given by $(C,\alpha)_{conv}$ and $(C,\alpha)_{tex}$.

We utilize the synthesis of additional channels for generating the alpha channel: In addition to the color input exemplar, an additional 2D texture is defined that encodes opacities. After the texture morphing has finished, the texture coordinates stored in the final synthesis pyramids can be used to look up pixels in this additional texture. Perception of the geometry of opaque texture elements, rather than the textured surface, can be improved by using the gradient of the scalar field $g(P) = f(P) \cdot a(P)$ as illumination function. Here f denotes the data set and a is a function that returns the geometry-defining alpha value of the solid texture at a sample point P.

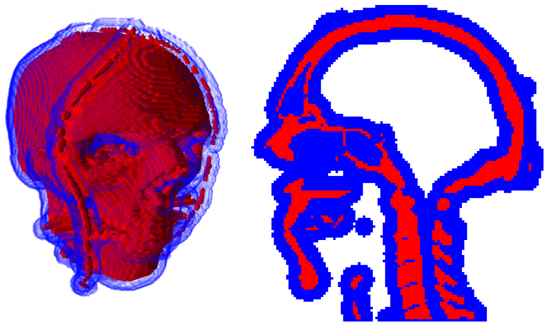
Figure 1: A binary shell mask (blue) with dilation radius $r_d = 5$ for a binary object mask (red).

5.2 Binary Shell Masks

In DVR the visualized data typically contains large regions that are of no interest and hence defined by the transfer function as fully transparent. Since texture synthesis and morphing is a time consuming process it is desirable for texture-enhanced DVR to generate solid textures only where they are required.

We realize partial texture synthesis and morphing by integrating a *binary object mask* into the basic 3D texture morphing algorithm. This binary object mask **B** is of the same size as the target texture cube **S**, and encodes which voxels of **S** are textured. Because our texture synthesis algorithm uses neighborhood matching and a multi-resolution approach, the binary object mask **B** is *dilated* by a user-defined dilation radius r_d (figure 1). In order to compute the shortest Euclidean distance of each voxel to the binary object mask, we adopt a C++ implementation by (Coeurjolly, 2003) that realizes a linear-time algorithm for computing the *3D squared Euclidean Distance Transform* (Meijster et al., 2000). In practice we found that a dilation radius $r_d = 10$ yields high quality textures while still resulting in significant time savings.

6 RESULTS AND DISCUSSION

We compared conventional transfer functions with texture transfer functions (figure 2) and found that for opaque textured layers, such as the bone layer in the figure, texture transfer functions enable similar perception of features. The geometry of semi-transparent layers, such as the skin layer of the nose, is harder to perceive than for conventional transfer functions.

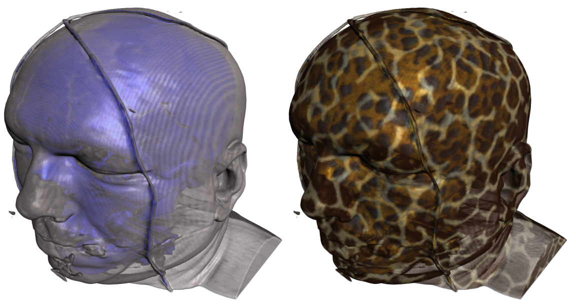
Figure 2: Comparison of renderings using the color of a conventional color-opacity transfer function (left) and of a texture transfer function (right). The gradient magnitude of the data serves as threshold for the opacity.

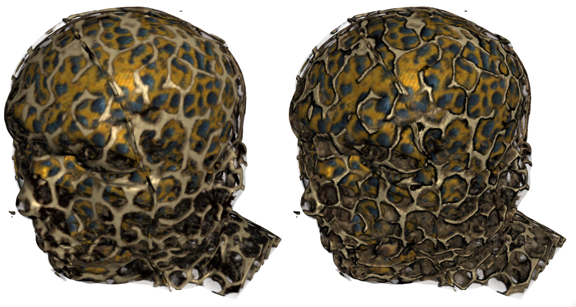
Figure 3: Screen-door transparency effects for overlapping layers. The mesh structure of the skin texture is fully opaque, whereas the regions in between are almost completely transparent. Left: Conventional illumination function using the gradient vectors for the object geometry only. Right: Improved illumination function taking the geometry of texture features into account.

However, the texture properties can now encode additional information such as fiber direction, cellular abnormalities etc. Similar as for color transfer functions gradient based shading is required for material boundaries. Since textures are heterogeneous in 3D we found it useful to additionally threshold the opacity with the gradient magnitude.

Perception of nested textured regions is difficult. Figure 3 demonstrates that perception can be improved by using screen-door transparency effects. The mesh structure of the texture used for the skin layer is defined to be fully opaque, whereas the regions in between are almost completely transparent. As a result, the features of the outer texture are not blended with the underlying texture of the bone. Note how the perception of texture features improves when the illumination function takes into account the geometry of texture features.

Figure 4 extends the previous visualization by additionally encoding the magnitude of the scalar gradient in colors. The left image shows a conventional rendering using only the color-opacity transfer function. On the right, texture-enhanced DVR is used to

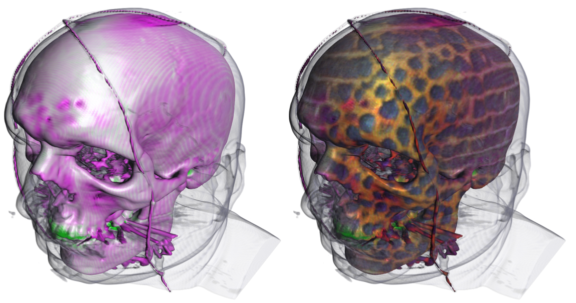
Figure 4: Using color and texture transfer functions together for encoding different information. The example encodes the magnitude of the scalar gradient in color, ranging from magenta (zero magnitude) over white to green (maximal magnitude) using color (left) and texture combined with colors using a multiplication operator (right).

encode the imaginary materials as before (using textures) as well as the gradient magnitude. The two colors are combined using a multiplication operator: $(C,\alpha) = (C,\alpha)_{conv} \otimes (C,\alpha)_{tex}$.

Note that such a morphing of two textures within one material layer is common for multi-field data. For example, a cardiac MRI data set could be used to differentiate different tissues such as the heart muscle, blood and surrounding fat tissue. A PET or fMRI data set could then provide additional functional information such as myocardial strain (Wünsche and Young, 2003) or viable, stunned, hibernating and dead myocardium represented by (morphed) textures.

6.1 Comparison with Previous Work

In section 2 we discussed Patel et al.'s scalar texture transfer functions which are similar to our texture transfer functions. However, there are fundamental differences in the two concepts:

Patel et al. use 2D textures and map them onto cutting planes through pre-classified layers. In contrast, texture-enhanced DVR provides texture information throughout the volume in three dimensions. Also Patel et al. only employ simple alpha-blending in order to place textures on top of each other, which does not preserve coherence of texture features. In contrast, texture-enhanced DVR provides a sophisticated morphing operator that creates smooth transitions between texture features. Arbitrary texture channels in addition to color and opacity can be defined.

In order to compute 2D texture coordinates and to ensure a correct texture mapping, Patel et al. need a separate volume to store information for parameterizing the cutout planes. In our framework, the transformation functions $\Gamma_{1\ldots L}$ guarantee a correct conversion from sample point positions to texture coordinates. Note that the functions can also be used for defining texture scaling, which is explicitly defined by Patel et al.'s texture transfer functions.

Finally, Patel et al.'s framework only allows to linearly blend between the textured visualization and the standard visualization with color-opacity transfer functions. In contrast, the color combination operator supports arbitrary mixtures.

Besides the conceptual differences, Patel et al. do not present any formalism for texture transfer functions in contrast to our mathematical framework for texture-enhanced DVR.

7 CONCLUSION AND FUTURE WORK

We have presented a new methodology for direct volume rendering termed *texture-enhanced DVR*. The research was motivated by the limited capabilities of color and opacity to convey multiple variables and attributes of a volumetric data set. Conventional DVR techniques rely on color-opacity transfer functions that map input data to an $RGB\alpha$ tuple. Texture-enhanced DVR extends this process by enabling the use of textures for encoding additional information. The new technique seamlessly integrates into the existing DVR process, yet is extremely powerful and flexible.

We introduced a mathematical framework which extends the existing DVR framework and is consistent with the use of textures for polygon rendering and previous applications of textures for DVR. In order to represent smooth transitions between different materials we use a new GPU-compatible 3D texture synthesis and morphing technique. Additional efficiency is gained by defining binary shell masks from the texture transfer functions and only synthesizing / morphing the textures where they are required. We investigated techniques to improve the perception of multiple and partially transparent textures and presented guidelines for their application.

In future work we want to further improve the perception of nested textured layers and we want to include procedural texture generation methods to better represent directional properties. Most importantly we want to use real medical multi-dimensional and multi-field data sets to demonstrate the usefulness of this new methodology in practice.

REFERENCES

Coeurjolly, D. (2003). 3D squared Euclidean distance transform. *http://www.cb.uu.se/~tc18/code_data_set/Code/SEDT/index.html*.

Helgeland, A. and Andreassen, O. (2004). Visualization of vector fields using seed LIC and volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 10(6):673–682.

Kniss, J., Kindlmann, G., and Hansen, C. (2002). Multi-dimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):270–285.

Landy, M. S. and Graham, N. (2004). *The Visual Neurosciences*, chapter Visual Perception of Texture, pages 1106–1118. MIT Press, Cambridge, MA, USA.

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

Manke, F. (2008). Texture-enhanced direct volume rendering. MSc thesis, University of Auckland, New Zealand.

Manke, F. and Wünsche, B. C. (2008). A direct volume rendering framework for the interactive exploration of higher-order and multifield data. In *Proceedings of GRAPP 2008*, pages 199–206.

Manke, F. and Wünsche, B. C. (2009). Fast spatially controllable 2D/3D texture synthesis and morphing for multiple input textures. In *Proceedings of GRAPP 2009*. (accepted for publication).

Max, N. (1995). Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108.

Meijster, A., Roerdink, J. B., and Hesselink, W. H. (2000). A general algorithm for computing distance transforms in linear time. In *Proc. of the International Symposium on Mathematical Morphology and its Applications to Image and Signal Processing*, pages 331–340.

Patel, D., Giertsen, C., Thurmond, J., and Gröller, M. E. (2007). Illustrative rendering of seismic data. In *Proceedings of Vision Modeling and Visualization 2007*, pages 13–22.

Rezk-Salama, C., Hastreiter, P., Teitzel, C., and Ertl, T. (1999). Interactive exploration of volume line integral convolution based on 3D-texture mapping. In *Proc. of Visualization '99*, pages 233–240. IEEE Press.

Wenger, A., Keefe, D. F., Zhang, S., and Laidlaw, D. H. (2004). Volume rendering of thin thread structures within multivalued scientific data sets. *IEEE Transactions on Visualization and Computer Graphics*, 10(6):664–672.

Wünsche, B. C. and Lobb, R. (2004). The 3D visualization of brain anatomy from diffusion-weighted magnetic resonance imaging data. In *Proceedings of GRAPHITE 2004*, pages 74–83. ACM Press.

Wünsche, B. C. and Young, A. A. (2003). The visualization and measurement of left ventricular deformation using finite element models. *Journal of Visual Languages and Computing*, 14(4):299–326.