High-Level User Interfaces for Transfer Function Design with Semantics

Christof Rezk Salama, Maik Keller, and Peter Kohlmann

Abstract—Many sophisticated techniques for the visualization of volumetric data such as medical data have been published. While existing techniques are mature from a technical point of view, managing the complexity of visual parameters is still difficult for non-expert users. To this end, this paper presents new ideas to facilitate the specification of optical properties for direct volume rendering. We introduce an additional level of abstraction for parametric models of transfer functions. The proposed framework allows visualization experts to design high-level transfer function models which can intuitively be used by non-expert users. The results are user interfaces which provide semantic information for specialized visualization problems. The proposed method is based on principal component analysis as well as on concepts borrowed from computer animation.

Index Terms—Volume rendering, transfer function design, semantic models.

1 Introduction

Direct volume rendering techniques are the most effective methods for visualizing tomographic data in clinical scenarios. Apart from medical applications, volume rendering is of great importance in natural and computational science, industrial design, engineering, and many other application areas.

When looking at the variety of solutions which have been published throughout the years, it seems reasonable to assume that the volume rendering problem is solved. Indeed, the technical problems, which are mainly the evaluation of the underlying physical model in real-time as well as the memory management for large data sets, have been successfully overcome in the past. In practice and here especially in medical practice, however, the existing solutions are still not used as frequently as one would expect. There is no technical reason for this.

Many non-expert users, such as physicians and other scientists with only marginal knowledge of computer graphics, often experience difficulties in managing the complexity of visual parameters. The majority of direct volume rendering techniques interpret the scalar field as a participating medium which emits and absorbs radiative energy. Many users report problems involving the process of specifying optical properties for a given data set. Manual assignment is cumbersome and time-consuming. In complex cases the visible effect of parameter modifications is often hardly predictable, even for visualization experts. On the one hand, automatic approaches are often not flexible enough. They are difficult to adapt to a wide range of data sets. On the other hand, they are often not specific enough to account for the precise task that the user wants to perform. If automatic approaches fail to deliver satisfying results, non-expert users are often left alone.

Throughout our co-operation with clinical partners, we have performed the visualization of medical data together with the responsible physicians many times . In this context we have realized the lack of clear semantics in this process. The physicians often made suggestions like *"Try to make the vessels sharper!"*, *"Fade out the soft tissue!"* or *"Improve the contrast between skin and bone a little bit!"* Even for a person familiar with the underlying transfer function model and the respective editor, it is not always easy to figure out which modification to the primitives will yield the desired result.

- *Christof Rezk-Salama and Maik Keller are with the Computer Graphics and Multimedia Systems Group, University of Siegen, Germany, E-mail: {rezk, keller}@fb12.uni-siegen.de.*
- *Peter Kohlmann is with the Institute of Computer Graphics and Algorithms, Vienna University of Technology, Austria.*

Manuscript received 31 March 2006; accepted 1 August 2006; posted online 6 November 2006.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

With regard to multi-dimensional transfer functions, appropriate user interfaces are difficult to operate. Even if the parameters can easily be specified by moving the handles in the editor, the visual results of the modification are often hard to predict. The specification of a transfer function is still a trial-and-error process which is extremely time-consuming in practice.

The issue of difficulties arising from complex applications being used by non-expert users is not a problem that is unique to visualization. Other fields such as computer animation, for example, have already overcome such problems to a large degree. This paper investigates how concepts from computer animation can be applied to improve the usability of direct volume rendering applications. We introduce a high level semantic model with a simple user interface. This concept allows visualization experts to design transfer function models for specific application areas, which can then be used intuitively by non-expert users.

The remainder of this paper is structured as follows. In Section 2, we have put together relevant related work. In Section 3, we review concepts from computer animation, which were the source of inspiration for our user interface design. Section 4 introduces the theoretical basis of our semantic models. Section 5 proposes efficient ways to design semantic models in practical cases. In Section 6 we describe details of our implementation of the proposed concept. The results of our technique are discussed in Section 7. In Section 8, we draw conclusions and comment on future work.

2 Related Work

Many sophisticated techniques to solve the volume rendering integral in real-time have been proposed in the past, including the shear-warp algorithm [16], 3D texture slicing [2, 31], 2D texture mapping [19], pre-integration [4], GPU ray-casting [15, 22, 26, 15], and special purpose hardware [17]. A detailed overview of GPU-based volume rendering can be found in the book by Engel et al. [3]. The ideas described in this paper are independent of the specific implementation.

Most approaches to direct volume rendering for scientific purposes are based upon a simplified physical model of light transport. Light is assumed to travel along straight lines and this assumption allows us to integrate radiative energy along rays of sight. The mapping of scalar values to optical properties is called *transfer function*. Usually, the required optical properties are emission and absorption coefficients. Kniss et al. [10] propose a more elaborate model of light transport involving shadows and translucency. Their model requires the specification of additional physical quantities. The concepts described in this paper are independent of the set of properties required for image synthesis.

If the scalar value alone is not sufficient to derive the optical properties required for rendering, multi-dimensional transfer functions may

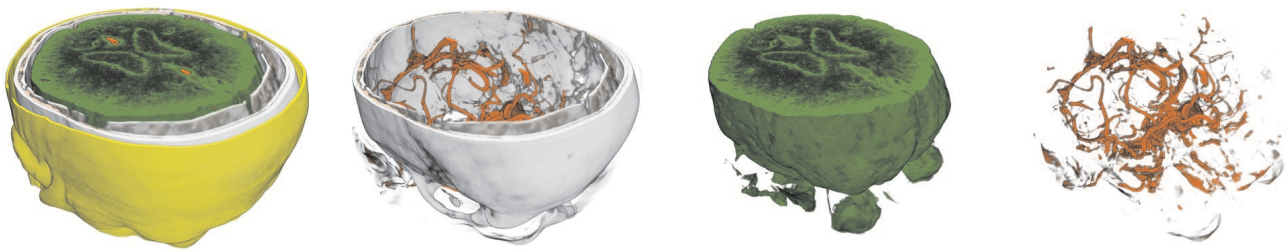

Fig. 1. Semantic models allow non-expert users to intuitively visualize volume data without knowledge about the visual parameters involved in the process of image synthesis.

be used as proposed by Kniss et al. [9]. The magnitude of the first and second order derivatives of the scalar field are frequently used to expand the transfer function domain. Vega et al. demonstrate the benefit of 2D transfer functions for visualizing blood vessels in CT angiography data [30]. Kniss et al. [12] use multi-dimensional transfer functions to classify co-registered multi-variate MRI data. At the bottom line, multi-dimensional transfer functions are highly superior to traditional 1D transfer functions. However, the complexity of parameter specification increases significantly with each additional dimension.

Automatic approaches to transfer function design can be categorized into image-driven or data-driven techniques. Most existing techniques are applicable for creating one-dimensional transfer functions only. Image-driven approaches analyze the information contained in images generated with different parameter settings and can be further divided into interactive evolution methods [13, 24, 27] as well as approaches which search for optimal settings based on objective quality measure (*inverse design* [25, 29, 7, 32]). He et al. [6] have developed a technique for semi-automatic transfer function generation, using stochastic search algorithms which are either controlled by manual thumbnail selection (like in [14]) or by an objective metric such as entropy, edge energy, and histogram variance. Although image-driven approaches represent a helpful aid for inexperienced users, they are not necessarily faster or more goal-directed than manual assignment.

Data-driven techniques analyze the volume data itself instead of analyzing generated images. The process of transfer function design is thus decoupled from the influence of image related parameters such as viewing position and pixel resolution. Fang et al. [5] and Sato et al. [23] do not generate transfer function tables as described above. Instead, they derive optical properties by applying 3D image processing operations directly to the volume data. Bajaj et al. [1] propose a data-driven technique which evaluates statistical information about area and gradient magnitude of the isosurfaces contained in the data. Tzeng et al. [28] utilize neural networks to derive transfer functions in an interactive process.

The most prominent data-driven technique was presented by Kindlmann and Durkin [8]. Their semi-automatic approach is capable of determining material boundaries within a given data set by evaluating statistical information about the first and second order directional derivatives. It is probably the most effective method which is currently available to visualize shapes and structures in an unknown volume data set by using a 1D transfer function.

In most practical cases, however, the user knows exactly which structures are contained in his data set and wants to visualize these structures of interest as fast as possible, without detailed knowledge of the rendering algorithm or the transfer function. For this purpose, we introduce an additional level of abstraction which completely hides the transfer function from the user by providing a limited set of semantic parameters.

3 Computer Animation

Our ideas to facilitate visual parameter assignment are based on concepts borrowed from computer animation. To explain these ideas, let us briefly consider the process of generating virtual characters in computer animation. In a typical production, there is a technical director who is responsible for creating the articulated model for each character, which can afterwards be controlled intuitively by the animator.

Each individual character has a set of expressions which he must be able to perform according to the underlying story board. For example, the character's face will probably be able to smile or frown and the animator will have high-level parameters to directly control such facial expressions.

From the point of view of the technical director, however, each facial expression consists of a combination of multiple low-level parameters such as the activation of specific facial muscles. Generating a smiling face, for example, will involve movement of the lips, the cheeks, the eyelids, and the eyebrows, which are all controlled by different pre-defined blend shapes (visemes [21]). Additionally, the jaw, which is controlled by the kinematic skeleton, will open slightly, and wrinkles, which are controlled by textures and bump maps, will become visible on the forehead.

In order to provide intuitive control for the animator, the technical director compiles combinations of the low-level parameters required for each facial expression into high-level parameters. In the modeling and animation package Alias Maya™, for example, this is possible by creating so-called *driven keys*, which are key frames that are specified with respect to an abstract parameter axis instead of the time axis. This way, the technical director creates semantic parameters such as "smile" or "frown", and hides the complex setup of low-level parameters from the animator.

Let us go back to transfer function design now and investigate how the described concepts can be adapted to this task. Our basic idea is that the visualization expert who is familiar with all the parameters involved in image generation will play the role of a technical director. In the following part of this paper we will explain effective techniques to hide the complexity of parameter assignment from the non-expert client.

4 Semantic Transfer Function Models

Regardless of its individual representation, a transfer function simply can be considered as a collection of parameters. At the lowest level of abstraction, a transfer function may be implemented as a simple lookup table. Each entry in this table can be considered a separate parameter.

User interfaces for transfer function specification often provide an additional layer of abstraction by introducing simple shapes as primitive objects such as boxes, ramps, and trapezoids. In the transfer function editor these primitives can be manipulated and moved directly on the screen. Examples of primitives in case of 2D transfer functions are the trapezoids suggested by Kniss et al. [9], and the paraboloid shapes introduced by Vega et al. [30]. The 2D primitives used in our system are displayed in Figure 2.

Each primitive has a set of parameters, such as the color and opacity values and the position of control points. Modifying the shape of the primitives results in a parameter change which directly influences the transfer function. Regardless of its representation and its dimensionality, we assume in the following that the parameters which represent an individual transfer function can be specified as an array of n floating

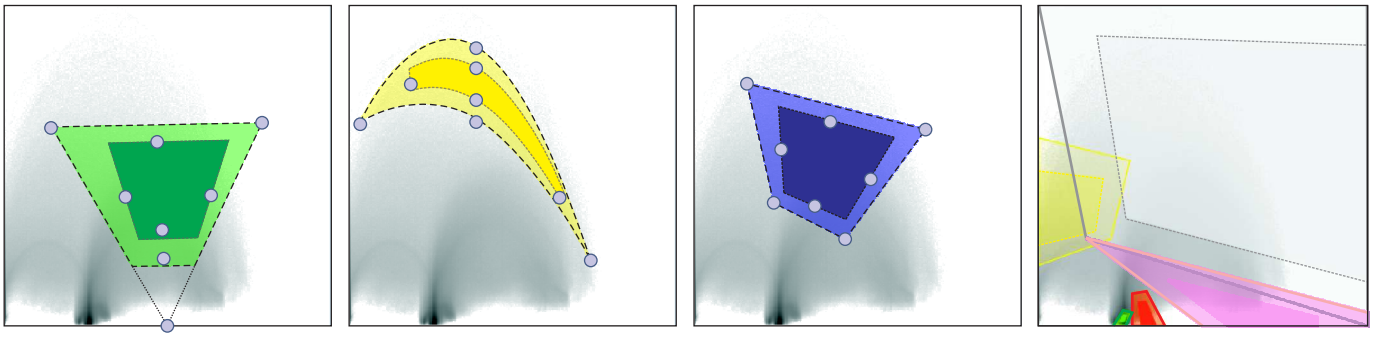

Fig. 2. The types of 2D transfer function primitives used in our system: Trapezoids (as in [9]), paraboloids [30] and quadrilaterals. The rightmost image shows the transfer function template used for CTA data. The 2D histogram for intensity and gradient magnitude is shown in the background

point values $\mathbf{p}$:

$$\mathbf{p} = (p_0, p_1, p_2, \ldots p_{n-1}) \in \mathbb{R}^n. \tag{1}$$

Most implementations convert the primitives into a color table representation before rendering. As an alternative, if programmable graphics hardware is used, transfer functions can be specified procedurally and evaluated at run-time, like the multi-dimensional Gaussian primitives proposed by Kniss et al. [11, 3].

As a basis for implementing semantics in our transfer function model we introduce a set of semantic parameters $\mathbf{s}$:

$$\mathbf{s} = (s_0, s_1, s_2, \ldots s_{m-1}) \in \mathbb{R}^m. \tag{2}$$

Generally, the number of semantic parameters m will be significantly smaller than the number of low-level parameters n, although this is not a necessary condition. Each semantic parameter $s \in \mathbb{R}$ has a predefined influence on the vector of low-level parameters $\mathbf{p}$. This influence is defined as a function $\mathbf{q}(s) : \mathbb{R} \mapsto \mathbb{R}^n$, and the final low-level parameter vector $\mathbf{p}$ is computed by summing up the influences of all semantic parameters:

$$\mathbf{p} = \mathbf{f}(\mathbf{s}) = \sum_{i=0}^{m-1} \mathbf{q}(s_i). \tag{3}$$

In general, the function $\mathbf{f} : \mathbb{R}^m \mapsto \mathbb{R}^n$ that maps semantic parameters to transfer function instances can be an arbitrary function. We have chosen a sum in order to keep the model intuitively understandable.

Similar to the *driven keys* described in Section 3, the influence of a semantic parameter s_i is specified by a sequence of nodes $(\sigma_j, \hat{\mathbf{q}}_j)$ with $\mathbf{q}(\sigma_j) = \hat{\mathbf{q}}_j$ and piecewise linear interpolation:

$$\mathbf{q}(s) = \begin{cases} (\frac{\sigma_1 - s}{\sigma_1 - \sigma_0})\,\hat{\mathbf{q}}_0 + (\frac{s - \sigma_0}{\sigma_1 - \sigma_0})\,\hat{\mathbf{q}}_1, & \text{for } \sigma_0 \le s < \sigma_1 \\ (\frac{\sigma_2 - s}{\sigma_2 - \sigma_1})\,\hat{\mathbf{q}}_1 + (\frac{s - \sigma_1}{\sigma_2 - \sigma_1})\,\hat{\mathbf{q}}_2, & \text{for } \sigma_1 \le s < \sigma_2 \\ \vdots & \vdots \\ (\frac{\sigma_k - s}{\sigma_k - \sigma_{k-1}})\,\hat{\mathbf{q}}_{k-1} + (\frac{s - \sigma_{k-1}}{\sigma_k - \sigma_{k-1}})\,\hat{\mathbf{q}}_k, & \text{for } \sigma_{k-1} \le s \le \sigma_k \end{cases}$$

With some effort it might be possible for an experienced user to specify the keys $(\sigma_j, \hat{\mathbf{q}}_j)$ manually for a given data set. The semantic models we are interested in, however, should work for more than one specific data set. We aim to derive a semantic model which is applicable to different data sets, provided that they have been recorded with a similar tomographic sequence and with the same examination purpose in mind. This applies to most examination procedures in medical practice, and we will show some examples in Section 7.

For simple relationships such as the direct influence of a semantic parameter *visibility* on the opacity of a single transfer function primitive, for example, it is easy to specify influence keys manually. In more complex relationships, more sophisticated techniques must be used to find appropriate keys that yield the desired visual results.

5 Design of Semantic Models

We suggest that the semantic model is designed in a co-operation between a computer scientist and a medical doctor. Before starting to implement a semantic model, the designers should talk to the client user and consider what structures are of interest and which semantic parameters are required. The challenge is then to find an appropriate set of weights which yield the desired result when the semantic parameters are finally used to instantiate the transfer function according to Equation 3.

The basis of our approach to designing semantic models is a set of reference data. Ideally, this set should statistically represent the range of possible data sets for the desired examination purpose. In practise, however, this condition is hard to verify, and we suggest using as many data sets of a specific type as available.

The next step is to make a list of the relevant structures contained in the data. We call each relevant structure an *entity*. In the CT angiography example shown in Figure 1, the list of entities comprise *bone*, *skin*, *brain tissue* and *blood vessels*. Each entity is represented by one or more transfer function primitives. Based on these primitives a transfer function model is created which is used as template for manual adaptation.

In the following step the transfer functions template is adjusted for the reference data. Each of the reference data sets is loaded one after the other into the volume renderer. The structures in the 2D histogram are slightly different for each data set and the primitive editor is used to adapt the transfer function model to each individual instance. As a result we obtain an instance of the (low-level) parameter vector $\mathbf{p}[i]$ for each reference data set i.

In order to create semantic parameters, we analyze the set of instance vectors using principal component analysis (PCA).

5.1 Principal Component Analysis

Each instance of the parameter vectors $\mathbf{p}[i]$ can be interpreted as a point sample in the n-dimensional low-level parameter domain of the transfer function. In order to apply PCA [18], the components of the instance vector are interpreted as random variables with Gaussian probability distributions. The mean value vector $\overline{\mathbf{p}}$ is approximated by averaging the instance vectors:

$$\overline{\mathbf{p}} = \frac{1}{N} \sum_{i=0}^{N-1} \mathbf{p}[i], \tag{4}$$

and the covariance matrix is computed:

$$\mathbf{C_p} = \frac{1}{N-1} \sum_{i=0}^{N-1} (\mathbf{p}[i] - \overline{\mathbf{p}})\,(\mathbf{p}[i] - \overline{\mathbf{p}})^{\mathrm{T}}. \tag{5}$$

The principal components of the joint probability distribution are determined by calculating the *eigenvalues* λ_j and the *eigenvectors* $\mathbf{e}_j$ of the covariance matrix $\mathbf{C_p}$. Since $\mathbf{C_p}$ is symmetric and positive definite,

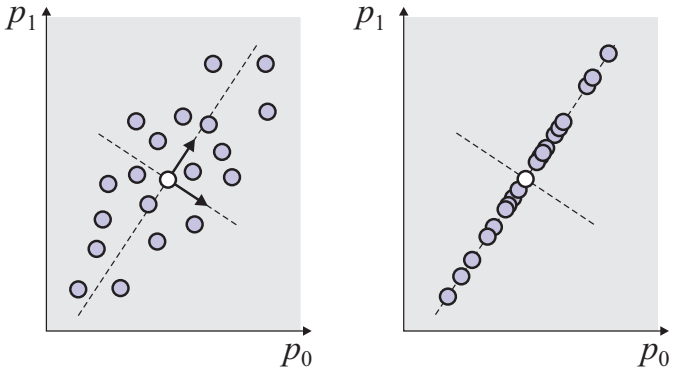

Fig. 3. Left: The instance vectors are point samples in the parameter space of the transfer function model. Principal component analysis is used to determine the axis of maximum variance. Right: Reducing the instance vectors to the first principal component approximates the data in a lower-dimensional subspace.

all the eigenvalues are real and non-negative. The unit-length eigenvectors are mutually orthogonal and represent a basis of the low-level parameter space.

The eigenvector corresponding to the largest eigenvalue is called *first principal component* and determines the axis of maximum variance of the joint probability distribution. Figure 3 shows a simple example for $n = 2$. If the instance vectors are represented by the first few principal components only, the possible range of the data is approximated in a lower-dimensional subspace. The quality of such an approximation is determined by the importance of the respective eigenvalues. The importance of a component axis can be determined by dividing the eigenvalue λ_j by the sum of eigenvalues:

$$I(\lambda_i) = \frac{\lambda_i}{\sum_{j=0}^{n-1} \lambda_j}. \tag{6}$$

Principal components with an importance close to zero can be safely omitted without a significant loss in accuracy. In our system, we use PCA to create basic semantic parameters for adapting a template to an individual data set. Afterwards, we build additional semantics as required for the specific visualization task.

5.2 Creating Parameters for Template Adaptation

Adjusting the transfer function template to an individual data set requires modification of the underlying primitives. These modifications, however, will most likely not affect all the low-level parameters, but only a limited subset. Additionally, the modified parameters are usually highly correlated. Principal component analysis can be used to determine a single parameter axis in the low-level parameter space from the reference data sets. This axis will approximate the modifications to the parameter vector $\mathbf{p}$. Throughout our experiments, we have found that in practical cases the importance of the first principal component is dominant and the transfer function template can be adjusted using a single parameter.

In this case we can create a semantic parameter *"adapt template,"* represented by a single slider in the user interface. The slider value can be changed by a non-expert user to adjust the transfer function to its individual data set without any knowledge of the underlying transfer function template at all. In order to determine the minimum and maximum values for the slider, we project all the instance vectors $\mathbf{p}[\mathrm{i}]$ onto the axis spanned by the first principal component $\mathbf{e}_0$. We do this by computing the n-dimensional dot-product:

$$\mu_i = \mathbf{e}_0 \cdot (\mathbf{p}[\mathrm{i}] - \overline{\mathbf{p}}). \tag{7}$$

The keys $(\sigma_j, \mathbf{q}_j)$ for the new semantic parameter can then be computed by

$$\sigma_0 = 0.0, \qquad \mathbf{q}_0 = \overline{\mathbf{p}} + \min(\mu_i) \cdot \mathbf{e}_0\,, \tag{8}$$

$$\sigma_1 = 1.0, \qquad \mathbf{q}_1 = \overline{\mathbf{p}} + \max(\mu_i) \cdot \mathbf{e}_0\,. \tag{9}$$

The values for σ_0 and σ_1 determine the range of the slider and can be chosen arbitrarily. The maximum and minimum computation ensure that the slider has no "dead space". Figure 4 illustrates a semantic parameter created for template adaptation of the blood vessels in the CTA example (see Section 7).

5.3 Creating Additional Semantics

Additional semantics such as *"sharpness"*, *"visibility"*, and "contrast" can be created by using the primitive editor to "learn" the necessary modifications of the underlying primitives. Since our mapping function $f(\mathbf{s})$ (see Equation 3) sums up the influences of all semantic parameters, the additional semantics must be specified with respect to the basic *"adapt template"* parameters explained in the previous section.

In order to create a semantic parameter that increases the contrast between different entities, for example, we again load each reference data set, one after the other, and use the basic parameter explained in the previous section to adapt the transfer function to the individual data. At this point we store the modified low-level parameter vector $\tilde{\mathbf{p}}[\mathrm{i}]$. Then we apply the modifications to the primitives that are necessary to increase the contrast and store the parameter vector $\tilde{\mathbf{p}}_{\text{contrast+}}[\mathrm{i}]$. Possible actions to increase the contrast are modifying the opacity slope at the border of a primitive or changing the brightness of the colors. Afterwards, we repeat this step, decrease the contrast, and store the parameter vector $\tilde{\mathbf{p}}_{\text{contrast–}}[\mathrm{i}]$ again.

We finally calculate the difference vectors

$$\delta_{\text{contrast+}}[\mathrm{i}] = \tilde{\mathbf{p}}_{\text{contrast+}}[\mathrm{i}] - \tilde{\mathbf{p}}[\mathrm{i}] \tag{10}$$

$$\delta_{\text{contrast–}}[\mathrm{i}] = \tilde{\mathbf{p}}_{\text{contrast–}}[\mathrm{i}] - \tilde{\mathbf{p}}[\mathrm{i}] \tag{11}$$

and the first principal components $\hat{\delta}_{\text{contrast+}}$ and $\hat{\delta}_{\text{contrast–}}$ as described in the Section 5.1. We can now define a new semantic parameter *"contrast"* with the following nodes:

$$\sigma_0 = -1.0, \qquad \mathbf{q}_0 = \hat{\delta}_{\text{contrast–}}\,, \tag{12}$$

$$\sigma_1 = 0.0, \qquad \mathbf{q}_1 = \vec{0}\,, \tag{13}$$

$$\sigma_2 = 1.0, \qquad \mathbf{q}_2 = \hat{\delta}_{\text{contrast+}}\,. \tag{14}$$

Additional keys may be inserted if necessary. Implementing a semantic parameter *"visibility"* is a good example in which multiple nodes can be used to improve the visual effect when the slider is moved. Simply turning down the opacity will make a rather homogeneous structure slowly disappear. In many cases, however, it is desirable to turn opacity down for low gradients first. This will turn a previously opaque object into a transparent shell, before it completely disappears. Such effects can easily be implemented using multiple keys for a semantic parameter.

There may be different modifications to the primitives that yield similar results. In order to decrease the visibility of a structure, for example, you may either turn down the opacity or reduce the size of the primitive. Our approach assumes that the creator of the semantic model uses the same action to achieve a specific task for all reference data sets.

6 Implementation Details

Our implementation of the technique described is built on top of a GPU-based volume rendering system in C++ and OpenGL which supports both 3D texture slicing [31] and 2D multi-texture based rendering [19]. The semantic models have been implemented for traditional 1D transfer functions and for 2D transfer functions based on intensity and gradient magnitude. All images in this paper were generated using 2D transfer functions and 12 bit volume data.

Transfer functions have been implemented with programmable graphics hardware and post-interpolative 1D or 2D dependent texture lookups. With regard to 2D transfer functions, the magnitude of the gradient vector is calculated using a multi-level technique. Each level is created by downsampling the previous level (i.e. averaging eight

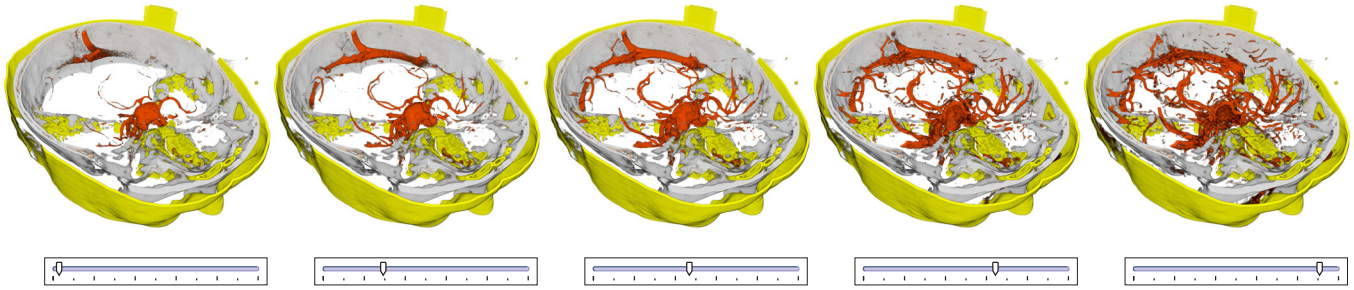

Fig. 4. Example of template adaptation to a data set which was not contained in the reference training set. The images have been generated by incrementally moving the *"adapt template"* slider for the blood vessels.

voxels). The gradient magnitude is calculated for each level separately. Afterwards, each level is upsampled to the resolution of the previous level and averaged with it. This technique results in smooth gradient data without the necessity of additional filtering. The gradient magnitude data is uploaded to the GPU memory at 16 bit resolution.

The different types of 2D primitives supported by our system are shown in Figure 2, and comprise

- Quadrilaterals with 16 parameters, 8 values for the (x,y)-coordinates of the four vertices, 4 parameters for the color and opacity (RGBA) of the primitive, and 4 parameters for the opacity slopes at the borders.
- Trapezoid primitives with 14 parameters, one value for the x-position of the base vertex, 4 values for the (x,y)-coordinates of the two upper vertices, and one parameter for the shifting the lower base line. The parameters for RGBA and the opacity slopes are the same as in the quadrilaterals.
- Parabolic primitives with 15 parameters, 6 parameters for the (x,y)-coordinates of the control points for the upper arc and one parameter for the lower indent. The parameters for RGBA and the opacity slopes are the same as in the quadrilaterals.

All parameters with the exception of RGBA can be modified by moving the control points shown in Figure 2. The system can be easily expanded by implementing additional primitives.

Modifications to the semantic parameters are immediately mapped back to the low-level parameters of the primitives. The 2D transfer function table is then generated by rendering the shape of the primitives directly into an off-screen render target which is finally bound as a 2D dependent texture for volume rendering. It is important to account for the correct composition of overlapping primitives. In our implementation, we blend the RGB values based on their opacities and use a maximum operation for opacity A. The system was tested on an ATI Radeon X800 XT and an Nvidia Geforce FX 6800 graphics board.

7 Results

Our first application example is the visualization of CT angiography (CTA) data, collected within clinical practice at the Department of Neuroradiology at the University of Erlangen-Nuremberg. This collection of data was acquired for the purpose of operation planning for the treatment of intracranial aneurysms.

All image data has been recorded with a Siemens Somatom Plus 4 spiral-CT scanner. During data acquisition, non-ionic contrast agent (100ml) has been applied in all cases. The delay time was chosen with respect to the circulation time for each individual patient. The resolution of the slice images is fixed at 512×512 with 12bits per voxel. The number of slice images for each individual data set varies between 120 and 260. Twenty different data sets were used as reference data for constructing the semantic model. Five additional data sets were used to evaluate the model. Our experiments have shown that the resulting semantic model does not change significantly, if different data sets from the collection are chosen as a reference set. Examples of the reference data are shown in Figure 6.

The structures found in the 2D histogram vary considerably among the data sets, mainly due to the different fields of view during data acquisition. A transfer function template was created accounting for the entities *bone structures*, *brain/soft tissue*, *skin/cavities*, and *blood vessels*. The template is shown in the rightmost image of Figure 2, with one primitive for the vessels (red), the brain (green), the skin (yellow), respectively, and two primitives for the bone (white and pink). The bone structures are represented by two separate primitives to improve their visual appearance.

After the template has been adapted to all reference data sets, the transfer function is split into groups of primitives that belong to the same entity. Principal component analysis is performed separately for each entity. This is necessary due to the different scales of the primitives. As can be seen in the template in Figure 2, the sizes of the primitives differ considerably for different entities.

The implementation of the techniques described in this paper has been evaluated using collections of real patient data sets from two different studies in clinical practice.

A rather large modification of the large white primitive (bone) may have a less significant visual effect to the final rendition than a subtle modification to a small primitive (vessels or brain). If PCA is per-

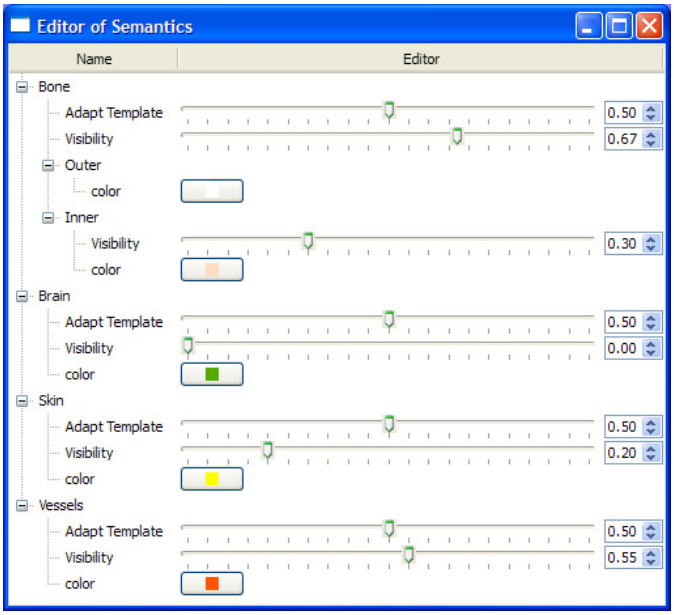

Fig. 5. The user interface created by the semantic transfer function model for CT angiography data. The numbers in the spin boxes represent the numerical values of the sliders.

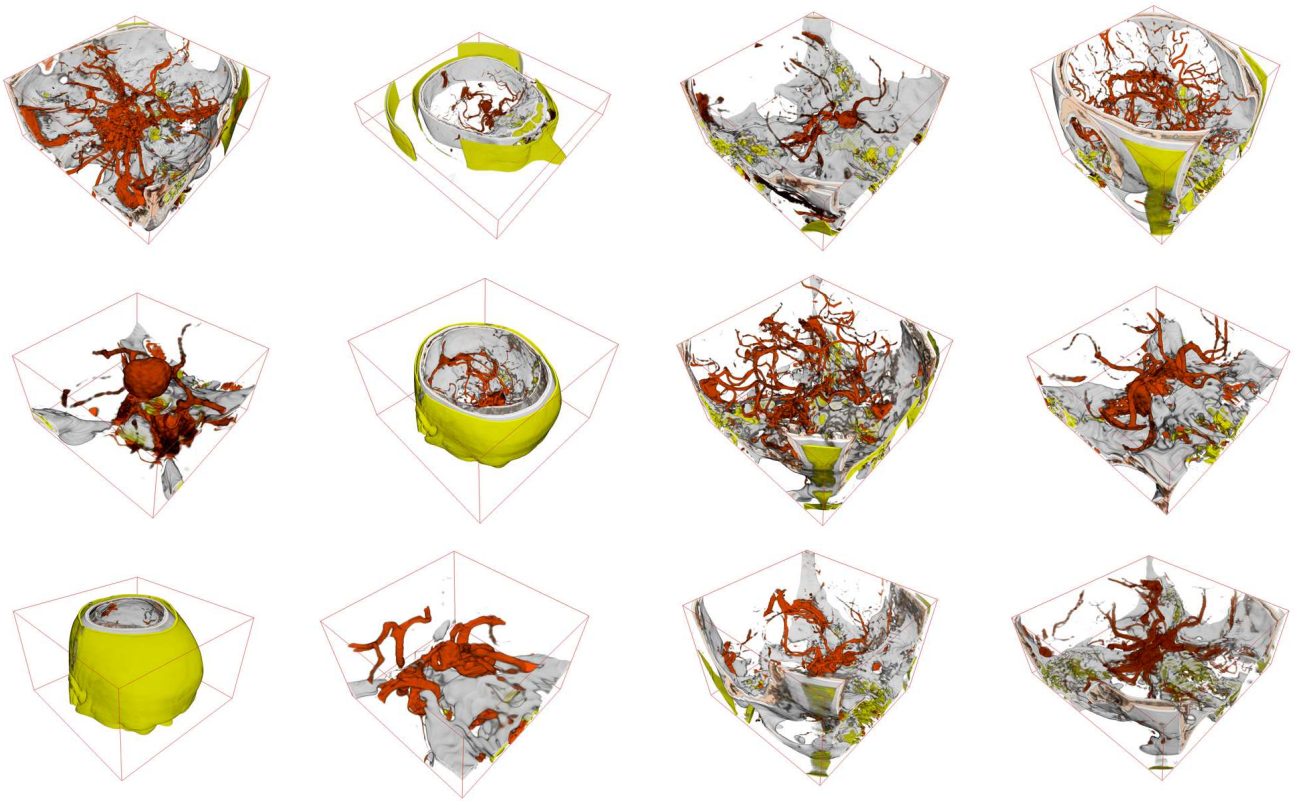

Fig. 6. CT angiography: 12 out of the 20 reference data sets from the clinical study on intracranial vessels

formed for the complete parameter vector containing all primitives, the subtle but important adjustments to the small primitives can easily be lost, because the covariance matrix is dominated by the large variance of other low-level parameters. If it is desirable to have one semantic parameter that adapts the whole transfer function, it is necessary to specify additional scaling factors for the individual components of the parameter vector, but we have found that this will have hardly predictable effects on the semantic model.

Performing PCA for each entitiy creates a semantic model with separate *"adapt template"* parameters for each structure of interest. The user interface generated from the underlying semantic model is shown in Figure 5. Additional semantic parameters have been implemented for *"color"* and *"visibility"*. The *"visibility"* parameters for vessels, brain, and skin simply specify the opacity of the respective primitive. For the bone structures, the visibility slider can be used to simultaneously fade out both primitives. The opacity of the inner bone structures (bone marrow, pink) can separately be controlled by an additional slider to enhance the visual appearance. For the CT angiography data, this set of parameters turns out to be completely sufficient for creating all the visual representations that the physician was interested in. Example images created for data sets which are not contained in the reference set are shown in Figure 7.

The second application scenario was the visualization of preoperative MRI data acquired for the planning of tumor resection in the brain, provided by the Department of Neurosurgery of the University of Erlangen-Nuremberg. The resolution of the slice images were fixed at 256×256, and the number of slices varies between 150 and 200. For the creation and evaluation of our semantic model, 10 MRI data sets were available. We chose seven data sets as reference sets, and three data sets for studying the results. Again, the resulting semantic model was almost independent of the choice of reference data sets. Example images are shown in Figure 8.

The image quality of the MRI data is limited due to the short period of time permitted for data acquisition in clinical practice. In consequence, there is significant noise inherent in the data. The entities chosen in this case were *"brain tissue"* and *"skin"*. Each entity was represented by a single primitive. Besides the semantic parameters for *"adapt template"*, *"color"*, and *"visibility*, an additional parameter *"sharpness"* was introduced. This parameter controls the sharpness or fuzziness of the brain surface and was implemented mainly by modifying the opacity slopes at the border. The effect of this parameter on the transfer function can be seen in Figure 9. Using this semantic model, the surgeon can intuitively analyze the pre-operative volume data during intervention planning.

7.1 Evaluation

We investigated the importance and the stability of the first eigenvector for each semantic parameter. For the CTA data, the axis spanned by the first principal component was stable if more than 12 data sets were used as reference sets, which means that the axis did not change significantly, if more than 12 different data sets were chosen. We considered a principal component as stable, if the dot product of the normalized axes from different evaluations did not fall below 0.9. It is worth noting that even for dot products of about 0.7, the subjective visual impression of the user was that resulting semantic models behaved in the same way.

The importance of the first principal component in the worst case was 0.85 for 12 reference data sets and 0.92 for 20 data sets. For less than eight data sets the importance of the largest eigenvalue dropped below 0.6. It might also be interesting to know that the significance of the first principal component is slightly higher, if the manual template adaptation was performed by the same person for all data sets instead of several persons. It shows that the stability of the described approach depends on the template adaptation strategy of the designer, which might be considered a drawback of the technique.

For the MRI scenario, the importance of the first principal component was 0.6 in the worst case and 0.8 in the best case, due to the limited number of reference data sets. A higher number of training data sets would have been necessary for a detailed analysis. In general, the presented approach has the danger to over-parameterize or

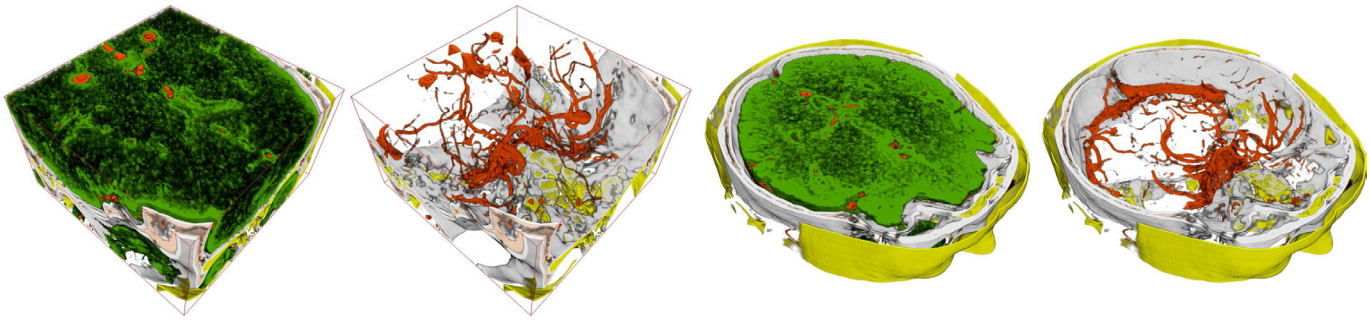

Fig. 7. The semantic model for CT angiography applied to two data sets, which were not part of the reference training set.

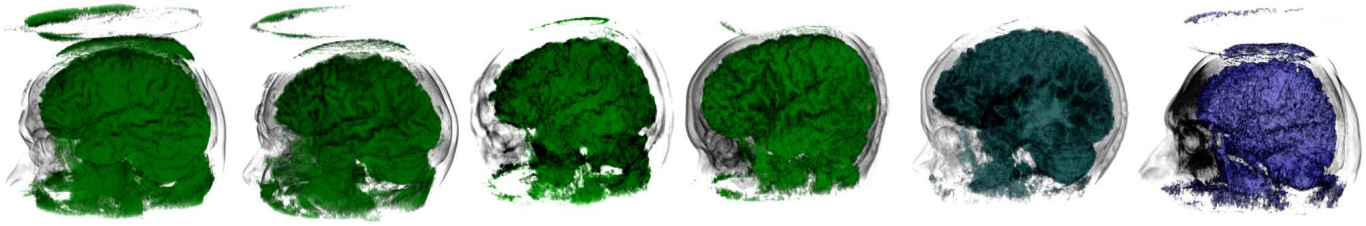

Fig. 8. Pre-operative MRI from Brain Surgery: The four leftmost images show 4 of the 7 reference data sets used in creating the semantic model. The two rightmost images show the semantic model adapted to data not contained in the reference data set. The entities have been adapted to the data and the color of the brain has been changed in the semantic model. (The contrast was slightly enhanced for the printout)

under-parameterize the parameter space of the transfer function. Although this is always kept under control of the designer, it might also be considered a drawback of the presented technique.

After the technical analysis, the semantic model was evaluated by uninvolved users. The users were presented with an unlabelled user interface and were asked to describe the effect of the slider changes. In most cases the user did properly describe the visual effect intended by the designer, such as ”visibility”, and ”sharpness”. The template adaptation parameter, however, was often inadequately described by the users. A detailed usability study in clinical practice is still pending. Up to now, the user interface was presented to a radiologist. He was asked to perform data visualization for a typical clinical examination. His feedback to the high-level user interface was very positive. He stated that he would prefer the simplified user interface to the primitive-based editors he was accustomed with. He appreciated the template adaptation parameter for manual tweaking, although he emphasized that an automatized initial setup would also be desirable. We will investigate if the technique described in [20] can be used to achieve this task in a future version.

As a result, we have found that in order to provide a good user interface it is not necessary to account for all possible kinds of modifications to the low-level parameters. There is a considerable amount of redundancy in a low-level transfer function model. And although we are able to create a semantic model easily with a large number of parameters, the non-expert users are helped much more, if we restrict the user interface to a limited number of necessary parameters. Throughout our experiments we have found that this is possible without a significant loss of flexibility.

8 Conclusion and Future Work

We have presented a framework for implementing semantic models for transfer function assignment in volume rendering applications. We have demonstrated that semantic models can effectively be used to hide the complexity of visual parameter assignment from the non-expert user for a specific examination purpose.

The concepts described are not restricted to transfer function design. They can be used to provide intuitive user interfaces for different kinds of visualization tasks. Additional parameters such as edge properties, which are frequently used in non-photorealistic rendering, and material properties for local illumination and translucent rendering can also be implemented as semantic parameters.

The assumption of Gaussian distribution for the low-level parameters was made to motivate the use of PCA. Gaussian probability distribution is completely defined by the mean value and the variance. First and second order statistics are sufficient for analyzing Gaussian densities, since the higher-order moments do not carry additional information. It might be worth investigating if analysis using higher-order statistics such as independent component analysis can be used to derive non-linear semantic parameters. Such an investigation, however, would require enormously large reference data sets to estimate parameters reliably.

Acknowledgements

The CT angiography data was generously provided by Dr. Bernd Tomandl, now at the Klinik of Neuroradiologie, Bremen, Germany. The MRI data is courtesy of Dr. Christopher Nimsky from the Department of Neurosurgery, University of Erlangen-Nuremberg, Germany. A very special thank-you is due to Kathrin Ohrndorf for proof-reading and speaking the accompanying text for the demonstration video.

References

[1] C. Bajaj, V. Pascucci, and D. Schikore. The Contour Spectrum. In *Proc. IEEE Visualization*, 1997.

[2] B. Cabral, N. Cam, and J. Foran. Accelerated Volume Rendering and Tomographic Reconstruction using Texture Mapping Hardware. In *Proceedings of IEEE Symposium on Volume Visualization*, pages 91–98, 1994.

[3] K. Engel, M. Hadwiger, J. Kniss, C. Rezk-Salama, and D. Weiskopf. *Real-Time Volume Graphics*. AK Peters, 2006.

[4] K. Engel, M. Kraus, and T. Ertl. High-Quality Pre-Integrated Volume Rendering Using Hardware-Accelerated Pixel Shading. In *Proceedings of ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware*, 2001.

[5] S. Fang, T. Biddlecome, and M. Tuceryan. Image-Based Transfer Function Design for Data Exploration in Volume Visualization. In *Proc. IEEE Visualization*, 1998.

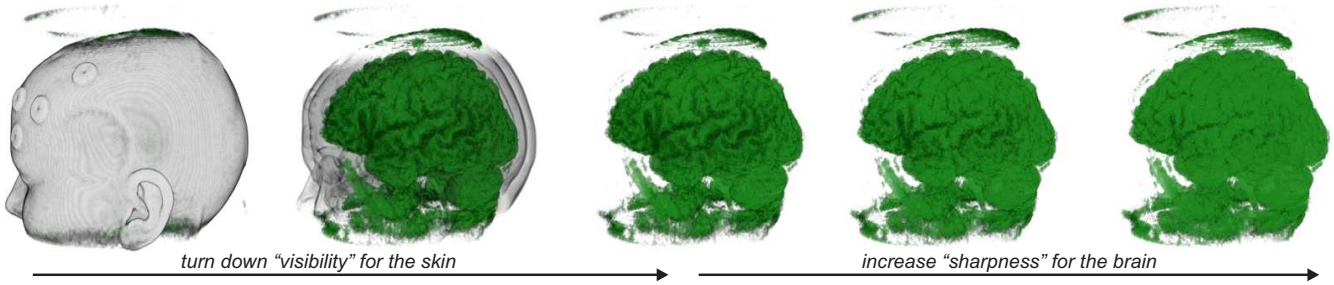

Fig. 9. Pre-operative MRI from Brain Surgery: The three image from the left demonstrate different settings for the *"visibility"* parameter of the skin. The three images from the right show the influence of the additional semantic parameter *"sharpness"*. (The contrast was slightly enhanced for the printout)

[6] T. He, L. Hong, A. Kaufman, and H. Pfister. Generation of Transfer Functions with Stochastic Search Techniques. In *Proc. IEEE Visualization*, 1996.

[7] J. Kawai, J. Painter, and M. Cohen. Rapidoptimization – Goal-Based Rendering. In *Proc. SIGGRAPH*, 1993.

[8] G. Kindlmann and J. Durkin. Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering. In *IEEE Symposium on Volume Visualization*, 1998.

[9] J. Kniss, G. Kindlmann, and C. Hansen. Interactive Volume Rendering using Multi-dimensional Transfer Functions and Direct Manipulation Widgets. In *Proceedings of IEEE Visualization*, pages 255–262, 2001.

[10] J. Kniss, S. Premoze, C. Hansen, and D. Ebert. Interative Translucent Volume Rendering and Procedural Modeling. In *Proceedings of IEEE Visualization*, 2002.

[11] J. Kniss, S. Premoze, M. Ikits, A. Lefohn, C. Hansen, and E. Praun. Gaussian Transfer Functions for Multi-Field Volume Visualization. In *Proc. IEEE Visualization*, 2003.

[12] J. Kniss, J. Schultze, U. Wössner, P. Winkler, U. Lang, and C. Hansen. Medical applications of multi-field volume rendering and vr techniques. In *Proc. Eurographics/IEEE TCVG Symposium on Data Visualization*, 2004.

[13] S. Kochhar. A Prototype System for Design Automation via the Browsing Paradigm. In *Proc. Graphics Interface*, 1990.

[14] A. König and E. Gröller. Mastering Transfer Function Specification by Using VolumePro Technology. In *Proc. Spring Conference on Computer Graphics*, 2001.

[15] J. Krüger and R. Westermann. Acceleration Techniques for GPU-based Volume Rendering. In *Proceedings of IEEE Visualization 2003*, pages 287–292, 2003.

[16] P. Lacroute and M. Levoy. Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation. In *Proceedings of ACM SIGGRAPH*, pages 451–458, 1994.

[17] H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer, and L. Seiler. The VolumePro real-time ray-casting system. In *Proceedings of ACM SIGGRAPH*, pages 251–260, 1999.

[18] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 2nd edition, 1992.

[19] C. Rezk-Salama, K. Engel, M. Bauer, G. Greiner, and T. Ertl. Interactive Volume Rendering on Standard PC Graphics Hardware Using Multi-Textures and Multi-Stage Rasterization. In *Proceedings of ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware*, 2000.

[20] C. Rezk-Salama, P. Hastreiter, J. Scherer, and G.Greiner. Automatic Adjustment of Transfer Functions for 3D Volume Visualization. In *Proc. Vision, Modeling and Visualization (VMV)*, 2000.

[21] K. Ritchie, J. Callery, and K. Riri. *The Art Of Rigging, Volume 1*. Alias conductors program. Cg Toolkit, 2005.

[22] S. Röttger, S. Guthe, D. Weiskopf, and T. Ertl. Smart Hardware-Accelerated Volume Rendering. In *Procceedings of EG/IEEE TCVG Symposium on Visualization VisSym '03*, pages 231–238, 2003.

[23] Y. Sato, C.-F. Westin, and A. Bhalerao. Tissue Classification Based On 3D Local Intensity Structures for Volume Rendering. *IEEE Transactions on Visualization and Computer Graphics*, 6, April 2000.

[24] K. Sims. Artificial Evolution in Computer Graphics. In *Proc. SIGGRAPH*, 1991.

[25] K. Sims. Evolving Virtual Creatures. In *Proc. SIGGRAPH*, 1994.

[26] S. Stegmaier, M. Strengert, T. Klein, and T. Ertl. A Simple and Flexible Volume Rendering Framework for Graphics-Hardware-based Raycasting. In *Proceedings of the International Workshop on Volume Graphics '05*, pages 187–195, 2005.

[27] S. Todd and W. Latham. *Evolutionary Art and Computer Graphics*. Academic Press, 1992.

[28] F.-Y. Tzeng, E. Lum, and K.-L. Ma. A Novel Interface for Higher-Dimensional Classification of Volume Data. In *Proceedings of IEEE Visualization*, pages 505–512, 2003.

[29] M. van de Panne and E. Fiume. Sensor-Actuator Networks. In *Proc. SIGGRAPH*, 1993.

[30] F. Vega 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. In *Proceedings of SPIE Medical Imaging*, 2004.

[31] O. Wilson, A. V. Gelder, and J. Wilhelms. Direct Volume Rendering via 3D-textures. Technical Report UCSC-CRL-94-19, Univ. of California, Santa Cruz, 1994.

[32] A. Witkin and M. Kaas. Spacetime Constraints. In *Proc. SIGGRAPH*, 1988.