IEEE/ EG Symposium on Volume and Point-Based Graphics (2008)
H.- C. Hege, D. Laidlaw, R. Pajarola, O. Staadt (Editors)

Stroke-Based Transfer Function Design

T. Ropinski, J. Praßni, F. Steinicke and K. Hinrichs

Department of Computer Science, University of Münster, Germany

Abstract

In this paper we propose a user interface for the design of 1D transfer functions. The user can select a feature of interest by drawing one or more strokes directly onto the volume rendering near its silhouette. Based on the stroke(s), our algorithm performs a histogram analysis in order to identify the desired feature in histogram space. Once the feature of interest has been identified, we automatically generate a component transfer function, which associates optical properties with the previously determined intervals in the domain of the data values. By supporting direct interaction techniques, which are performed in the image domain, the transfer function design becomes more intuitive compared to the specification performed in the histogram domain. To be able to modify and combine the previously generated component transfer functions conveniently, we propose a user interface, which has been inspired by the layer mechanism commonly found in image processing software. With this user interface, the optical properties assigned through a component function can be altered, and the component functions to be combined into a final transfer function can be selected.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and RealismSubjects: Color, shading, shadowing, and texture

1. Introduction

Transfer functions are used to map intensity values to optical properties. In most application domains this mapping is specified in such a way that features of interest are visually more prominent, while less interesting features are visually less prominent or even masked out. Although many techniques for the specification of both 1D and 2D transfer functions have been proposed, until now there exists no widely accepted intuitive way. Manual setup of transfer functions is still a difficult task, which is very time-consuming and error-prone. Since achieved results are hard to reproduce, especially for 2D transfer functions manual setup by domain users is not practical. To cope with this problem, a couple of semi-automatic methods have been proposed. However, these methods are sometimes not intuitive, i. e., the resulting transfer functions do not match the intentions of the users, and they still require time for adaptation. Fully automatic transfer function specification does not improve the situation a lot, since these algorithms are neither flexible [PLB*01], nor are their results reliable for arbitrary data sets from different domains.

In this paper we propose interaction concepts for specifying transfer functions, which have been motivated by the work with our medical partners. When collaboratively adapting visualizations, the physicians had rather high level requests describing the desired visualization. For example, they asked to visually emphasize certain features of interest by changing their color or degree of opacity, or to add/remove features from the visualization. To depict the features of interest, they were often pinpointing to the respective position on the screen, while the histogram, commonly used for transfer function specification, was not in their focus. Besides this image-based way of thinking, it is also noteworthy that the physicians do not think of a transfer function as one holistic function, but as a list in which each entry is associated with a feature having assigned a color and an opacity. This perspective, which is similar to the component function approach presented by Castro et al. [CKLG98], does not match with the commonly used transfer function editors found in most medical applications. With these editors the transfer function setup is performed in the histogram domain by adding keys with associated color

© The Eurographics Association 2008.

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

and opacity. To obtain the transfer function, the program interpolates between these keys.

With the interaction metaphor proposed in this paper, we try to support the approach of the physicians described in the previous paragraph. By drawing one or more strokes near the silhouette of the feature of interest, our algorithm automatically generates a 1D component transfer function [CKLG98] for the identified feature. In order to be able to draw strokes along a silhouette, the feature obviously needs to be visible in the rendering. Therefore, we apply an initial black/white ramp transfer function and support windowing as known from medical workstations. When applying the windowing, the user can visually emphasize different features sequentially and is thus able to acquire information about the target data set. When the desired component transfer function has been generated, the user may control its optical properties. To do so, a user interface is proposed, which is inspired by the layer concept frequently found in image processing software. With this user interface, the color as well as the opacity of each identified component function can be altered conveniently. Furthermore, the component functions contributing to the final transfer function can be selected. When changing this selection or the optical properties, the rendering is updated on-the-fly in order to provide immediate visual feedback. While the combination of these interaction concepts allows to define 1D transfer functions easily, the proposed system is still subject to the known limitations of 1D transfer functions, especially it is not possible to separate features of interest which overlap in the data domain.

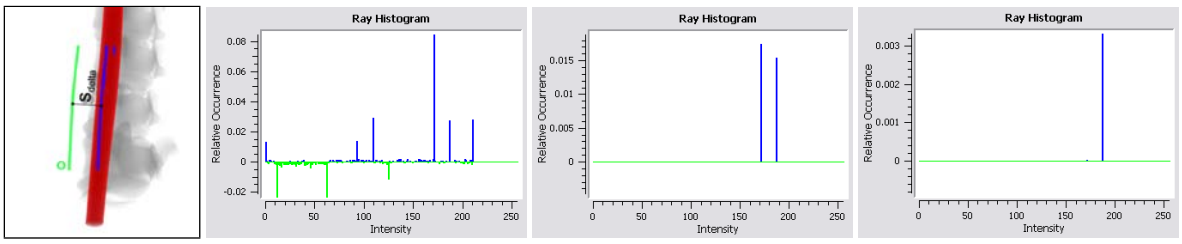
2. Related work

Transfer function specification has been an ongoing research topic in the past. Component functions, as described by Castro et al. [CKLG98], introduce a higher level of abstraction into the transfer function design process to allow a more intuitive transfer function specification. In medical visualization such an abstraction would be the characterization of different tissue types. To intuitively extract these features, component functions can be defined over different ranges of data values and finally be combined to the transfer function. This abstraction is similar to the material percentage volumes described by Drebin et al. [DCH88]. A material percentage volume can be generated through classification, and assigns the percentage of a material present at each voxel's location. Based on the material percentage volumes, composite volumes are generated for rendering by multiplying the material percentage with the property values assigned to that material. While Castro et al. [CKLG98] also discuss some user interface concepts for managing component functions, König and Gröller have gone further and propose an image-centric user interface for transfer function specification [KG01]. Similar to the component function approach developed by the same authors, they also choose a set of intervals in the data domain to which they assign colors, before they combine the resulting component functions into a single transfer function by considering user-specified opacities. In contrast to the stroke-based approach presented in this paper, their approach for specifying data ranges is an image-centric approach, which is inspired by the design gallery metaphor [MAB*97]. The layer analogy exploited by the user interface described in this paper has also been used by Rezk-Salama et al. [RSKK06]. They have adapted a semantic model for a selected application case and use it to integrate domain-specific knowledge into the user interface in order to hide the complexity of visual parameter assignment from the non-expert user. Another layer concept has been described by Rautek et al. [RBG07]. They propose how to specify a mapping from several volumetric attributes to multiple illustrative visual styles by using rules that are evaluated with fuzzy logic arithmetic. However, they stick with the concepts known from traditional transfer function editors to specify the attribute ranges used in the given rules.

In the transfer function bake-off [PLB*01] different image-centric and data-centric approaches are compared. Our approach can be classified as a data-centric approach, which aims at abstracting from the histogram domain as much as possible. The technique by Kindlmann and Durkin [KD98] rated as most promising among those compared in the transfer function bake-off focusses on the extraction of boundary surfaces. It uses the data values and the first and the second directional derivatives for generating a transfer function semi-automatically. The method has been extended by Kniss et al. [KKH02] through widgets and dual-domain operations. Sereda et al. [SBSG06] also emphasize boundaries with a method based on LH histograms. They identify and display surface representations in histogram space and enable the user to assign optical properties to these surfaces. Lum et al. [LSM06] adapt so-called filter banks from signal processing in order to generate more complex transfer functions, which are especially efficient for classifying noisy features. Similar to the material approach presented by Drebin et al. [DCH88], they exploit a classification based on material percentages, which can be configured by using a parallel coordinate user interface.

Sketch-based techniques have the potential to classify features in a volumetric data set directly in image space instead of interacting within the histogram space. One reliable method is to work on a single slice of the volume and to specify the contour of the desired feature or region manually. Tzeng et al. propose a painting interface which allows the user to directly draw on 2D slices [TLM03, TLM05]. Based on this drawing a transfer function is generated and a 3D visualization is updated rapidly. Similarly, Huang and Ma [HM03] allow the user to draw on a 2D slice to initiate a region growing process. They show that a partial region growing can be sufficient to derive a 2D transfer function. Chen et al. [CSSM06] combine 3D sketching with region growing for segmentation. Initially, they specify a region of interest by using multiple sketches, then they show how

© The Eurographics Association 2008.



(a) (b) (c) (d)

Figure 1: *Comparison of difference histograms generated with different weighting paradigms. The goal was to detect the* vena cava *($i = 188$), which occludes the* spinal column *($i = 170$) in this artificial data set. The distance between the strokes is s_{delta} (a). No weighting allows no peak detection (b). Opacity weighting still results in a too high* 170*s peak (c). Visibility weighting helps to extract the correct peak (d).*

to segment features from this region of interest by seeded region growing. A similar approach is also described by Sherbondy et al. [SHN03]. Wu and Qu [WQ07] propose a framework for combining existing volume renderings by using genetic algorithms in order to find a transfer function, which allows to visualize all relevant structures visible in the source images. They also discuss how to modify existing transfer functions by using sketching. They determine the features of interest by comparing the sketched silhouette to those contours automatically detected in the rendering. Thus, in contrast to our technique, designing a new transfer function requires an initial transfer function having certain properties, i. e., features must be visually separable. Owada et al. [ONI05] presented an intuitive sketching interface for segmentation purpose. The user traces the contour of the 3D target region using a 2D free form stroke. By analyzing the gradient length along rays cast through the contour, the system is able to segment a plausible volumetric region specified by the stroke. Similar to our approach the system infers the depth information of the desired region automatically by analyzing the data. However, in contrast to their technique, our approach does not require an accurate manual contour tracing, nor a gradient along the specified segment.

3. Stroke-based volume classification

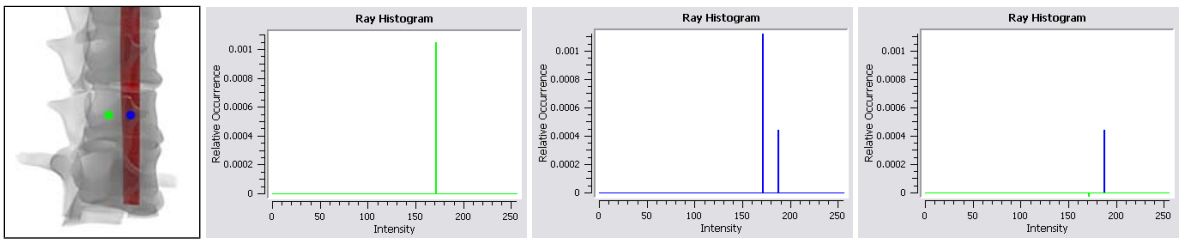
To make the specification of component functions more intuitive, we propose a stroke-based volume classification algorithm. By using this technique, a feature of interest can be selected by highlighting parts of its silhouette in image space with one or more mouse strokes. Since a silhouette may be rather complex, it is not necessary to sketch the whole silhouette, but only a small segment along it.

The stroke-based classification works as follows. Based on each drawn stroke two further strokes are generated, which are both parallel to the drawn one and positioned in the same distance on its opposite sides. The first one, the inner stroke, covers the feature of interest in image space, while the second one, the outer stroke, does not cover this feature (see Figure 1 (a)). Then our algorithm computes ray histograms for both of these strokes by casting rays through all points of each stroke. By comparing these histograms, we are able to identify the range of intensity values, which most likely represents the desired feature of interest, and we can generate a new component function associated with that intensity range. Besides generating new component functions, it is also possible to modify existing component functions. In the subtraction mode, the identified intensity range is removed from the intensity range of the current component function. This mechanism allows to further refine the classification results.

3.1. Stroke specification

When drawing the initial stroke, multiple mouse events at different positions are generated. For each position, where an event is detected, a control point is generated to specify the line strip representing the stroke. The distance s_{delta} between the inner and the outer stroke must be specified (see Figure 1 (a)). As mentioned above, the inner stroke covers the feature of interest, while the outer stroke should not cover it. It is obvious that the detection of different features may require an adaptation of s_{delta}. For instance, when drawing a stroke along the silhouette of a blood vessel, which might be a rather tiny structure in image space, s_{delta} can be rather small. In contrast, when detecting a larger scale feature, such as for instance the femoral bone, the value of s_{delta} can be larger. We have decided to let the user interactively specify s_{delta} by a horizontal mouse movement. After all strokes depicting a certain feature of interest have been drawn, the user can drag the mouse horizontally left resp. right in order to decrease resp. increase the value for s_{delta} interactively. If s_{delta} becomes negative, the inner and the outer stroke switch sides. Thus, in contrast to the work presented in [ONI05], our approach does not need an accurate tracing of the silhouette, and the stroke does not need to be drawn in a predefined direction. Especially the lower required accuracy makes the drawing process quite fast. Furthermore,

© The Eurographics Association 2008.



(a) (b) (c) (d)

Figure 2: *The computed ray histograms are shown for two different positions in the image (a). The left/green histogram contains the* spine*-peak only (b), while the right/blue histogram does contain two peaks (c). The desired peak can be extracted with the difference histogram (d).*

to deal with features having a tiny footprint in image space, we support defining a zooming area with the mouse.

3.2. Difference histogram generation

When the inner and the outer stroke have been defined, a difference histogram is generated and analyzed. This process starts with generating ray histograms for each control point on each of the two strokes. When all those ray histograms have been generated, pairwise difference histograms are calculated. Since the inner as well as the outer stroke are transformations of the initially drawn stroke, these difference histograms can be generated for their correlating control points. They are computed by subtracting the bin values of the outer ray histogram from those of the inner ray histogram. This results in a histogram having a positive peak for intensity values mainly present in the inner histogram and a negative peak for intensity values mainly present in the outer ray histogram. To get the final difference histogram, which represents the intensity distribution for the drawn stroke, we average over all difference histograms generated for all pairs of control points.

To facilitate the interpretation of the difference histogram, we have experimented with different paradigms for weighting the amount with which a voxel contributes to the ray histogram. A comparison of the three tested techniques is shown in Figure 1. Figure 1(a) shows the initial situation, the goal was to classify the opaque *vena cava*, which occludes parts of the semi-transparent *spinal column*. For explanatory purpose, this is an artificial 8 bit data set with known intensity values; the *vena cava* has an intensity value of 188, while the *spinal column* has an intensity value of 170. In Figure 1(b) the difference histogram is shown which has been generated without weighting the voxels, i. e., each voxel has the same contribution. As it can be seen, many peaks, positive as well as negative, contribute to the ray histogram. Since the assigned opacity values are not considered during the histogram generation, all intensities along a ray have a footprint in the final difference histogram. Although there is

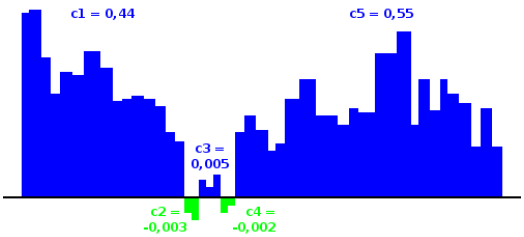


Figure 3: *A cutout of a difference histogram with five components (c_1-c_5). Positive bars (blue) indicate that the intensity values are mainly present in the inner histogram, while negative (green) bars indicate that the intensity values occur mainly in the outer histogram.*

a peak visible at 188, it is not possible to automatically identify this one as the desired one. Since the *spinal column* has a larger depth extent, more voxels having an intensity of 170 contribute to the histogram, and thus the 170s peak is reasonably larger. When incorporating the opacity of each voxel, all peaks but those of visible features can be omitted from the difference histogram (see Figure 1(c)). However, the highest peak still belongs to the *spinal column*. Although it has a much smaller opacity, the greater depth extent results in a higher peak. Therefore we have chosen to weight the voxels based on their visibility (see Figure 1(d)), which is initially modified through the windowing. By incorporating the opacities assigned through the current windowing, we evaluate the volume rendering integral to compute the visibility for each voxel along a ray, i. e., the fraction of its emission that reaches the camera. When computing the ray histograms, we weight each voxel based on the such computed visibility. As it can be seen, this makes the desired peak at 188 the most prominent one, despite there is still a small peak at 170 visible.

© The Eurographics Association 2008.

3.3. Histogram analysis

Figure 2 shows how a ray histogram, generated with visibility weighting, varies for different positions in image space. Positive values in the difference histogram indicate intensity values mainly present along the inner stroke, while negative values indicate that these intensities are mainly present along the outer stroke (see Figure 3). To determine the feature of interest, the highest peak needs to be detected. Unfortunately, features in real world data are given by an intensity range which consists of several adjacent positive peaks. We refer to a set of all adjacent positive or negative peaks as a histogram component. The cutout of the difference histogram shown in Figure 3 contains five components $c_1 - c_5$. While c_1, c_3 and c_5 are positive components, c_2 and c_4 are negative. A straightforward approach for identifying the component of interest, i. e., the one belonging to the feature of interest, would be to choose the one with the largest relative mass. However, in some cases this is not clearly defined, since different components with the same relative mass may exist. In the example shown in Figure 3, c_1 and c_5 have almost the same relative mass. Thus, when choosing the component c_5, the also significant component c_1 would be lost, since the components are split by the negative components c_2 and c_4. This occurrence of small negative components splitting significant positive components could be noticed especially when drawing unprecise strokes as well as when dealing with data sets having a higher bit depth or high noise occurrence. To handle this issue, we perform a connected component analysis in order to identify the set of components most likely representing the feature of interest. The goal is to merge the disconnected significant components into one component that reflects the feature of interest. To merge the disconnected components, we iterate over all positive components and test, whether they can be merged with their next positive neighbor. To evaluate this test, we introduce a merge factor m_{fac}. A merge is only possible if the negative component c_k separating two positive components c_i and c_j is sufficiently small, which we express by the following inequation:

$$mass(c_i) + mass(c_j) > -mass(c_k) \cdot m_{fac}. \qquad (1)$$

By applying this test, we successively merge all possible positive components. When no further merge steps are possible, we select the component with the largest mass, which represents our feature of interest. Thus, for the example shown in Figure 3 with $m_{fac} = 100$, initially c_1 and c_3 are merged to c_1c_3, since $0.44 + 0.005 > 0.003 \cdot 100$. Then, c_1c_3 is merged with c_5, and finally $c_1c_3c_5$ is selected as the component with the largest relative mass. For the data sets we have inspected, a value of m_{fac} below 500 gave good results. This shows that the negative peaks are rather small in comparison to the positive peaks. This must not be the case when dealing with data having a less good signal to noise ratio.

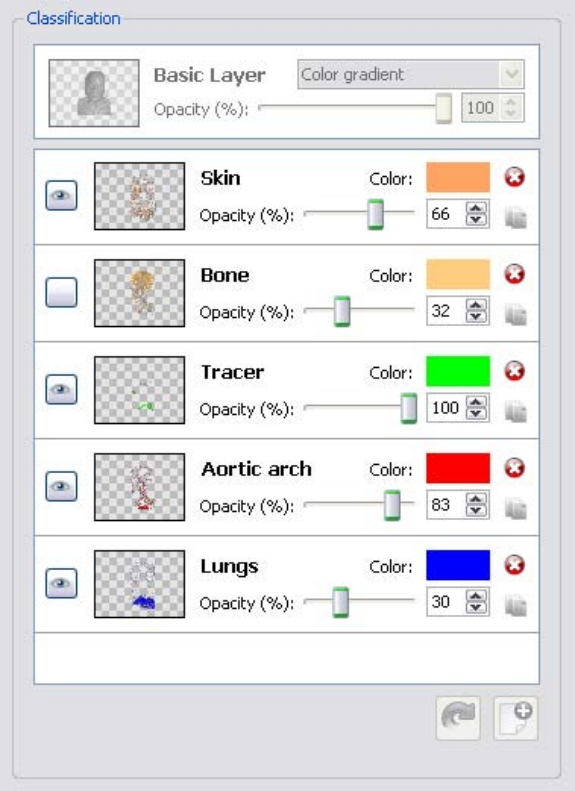


Figure 4: *Layer interface for modifying component functions. Each layer is specified by a name and some optical properties: color and opacity. The current appearance of the layer is depicted by a pictogram. By clicking the eye symbol, it is possible to toggle the visibility of a layer.*

4. User interface concepts

To modify and combine the generated component functions intuitively, we propose a user interface which has been inspired by the layer concept commonly found in image processing software. Within those applications, layers are used to constrain the user interaction to a subset of the visible features only. Furthermore, it becomes possible to undo even complex manipulations by discarding the appropriate layer. Similar interactions can be performed on component functions in order to make the manipulation more intuitive and to make reproduction easy. However, in contrast to the layer concept found in image processing software, the order of the component functions does not influence the result of the composition.

The proposed user interface contains a list of layers, representing the previously generated component functions. For instance, the user interface in Figure 4 shows the component functions used in Figure 7. To reduce the complexity of assigning optical properties, color and opacity are the only pa-

© The Eurographics Association 2008.

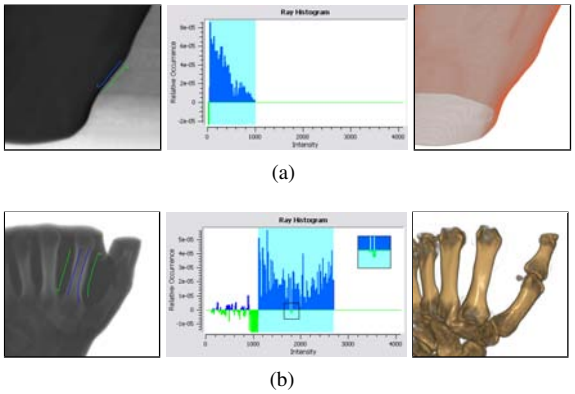
rameters the user has to control, via the color chooser and the opacity slider assigned to each layer. Based on these parameter settings, the associated component function is modified using a tent peak shape [KG01]. The user-selected color as well as the user-selected opacity are assigned to the center of mass of the intensity range, which is associated with the component function. To achieve a smooth color transition towards the boundaries of the covered data range, we exploit the HSV color space. We transform the user-selected color into the HSV space and divide the V value by two. The resulting darker color is assigned with zero opacity to the minimum and the maximum intensities lying in the associated intensity range. The remaining values are interpolated resp. extrapolated accordingly, which results in a simple tent-shaped component function. As Castro et al. [CKLG98] have described, it would be useful to visualize the impact of a single component function. Therefore we render preview pictograms, which are integrated into the user interface and are updated rapidly (see Figure 4). Once the transfer function designer is happy with the current layer, s/he may also adapt the associated name to make it more prominent.

By automatically combining all component functions into a final transfer function, it is possible to visualize a data set with all previously setup features. Currently, this combination is done by using a weighted average based on the opacity, where each color of the final transfer function is computed as follows:

$$C_{final}(j) = \frac{\sum_i C_i(j)\alpha_i(j)}{\sum_i \alpha_i(j)}. \qquad (2)$$

i is the index of the current transfer function and j an intensity in the data range. $C_i(j)$ and $\alpha_i(j)$ are the optical properties of the transfer function i for the intensity j. To get the opacities for the final transfer function, the opacities of the component functions can be summed. While this approach achieves good results, also more sophisticated techniques for combining the component functions could be integrated, for instance the one presented by Wu et al. [WQ07]. Similar to the layer processing of image manipulation applications, where layers can be set to be invisible, the combination of the component functions can be modified. By clicking the eye symbol (see Figure 4) a component function can be included into or excluded from the final transfer function to allow a comparison of alternative transfer function designs. Figure 4 shows the novice version of the user interface. To give expert users full control, the expert version also contains a traditional transfer function editor, which allows to adapt the component functions as well as the final transfer function freely.

Besides the layers associated with previously generated component functions, the user interface contains one layer, which is used when drawing the strokes and not considered during the combination of the component functions. This basic layer is associated with a ramp transfer function, which covers the whole intensity range and assigns zero opacity black to the minimum intensity value and white having a user-defined opacity to the maximum intensity value. By activating this layer and applying windowing, it becomes possible to show all features potentially contained in the data set. This process is inspired by the proceeding of a radiologist, who tries to identify features of interest by changing the windowing of a data set. Thus the transfer function designer can alter the voxel visibility through the basic layer until the features of interest become visible. To further support the visual distinction of intensity values, the color coding of this special transfer function can be switched from a gray value gradient to a color value gradient.



(a) (b)

Figure 5: *Applying our technique to a CT scan of a human hand. When classifying the skin only one component is significant (a); for the bone, several components are automatically merged to get the desired intensity range (b).*

5. Results

In Figure 5 the drawn stroke, the derived difference histogram as well as the resulting classification for different features of the hand data set are shown. Figure 5 (a) shows the results achieved when classifying the skin. As it can be seen, the histogram analysis reveals one connected component representing the feature of interest. Since its center of mass is slightly shifted to the left, the resulting tent-shaped component function is also slanted to the left. Figure 5 (b) shows the difference histogram generated when classifying the bone structures of the hand data set. In this case three components are potential candidates for the feature of interest. Since they are separated by negative components, our connected component analysis algorithm as described in Subsection 3.3 fixes the situation and helps to associate the merged component with the feature of interest.

By applying our technique to the baby data set shown in Figure 6, we could extract three layers in less than a minute. Once these layers have been specified, they can be changed and recombined easily. Figure 6 (a) shows the basic

© The Eurographics Association 2008.

(a) (b) (c) (d)

Figure 6: *A 1D transfer function has been generated for the baby data set by using three component functions, which have been created using the basic layer (a). The bone structures are visualized gray using a high opacity (b), red is assigned to the brain structures (c), and the skin is rendered semi-transparently (d).*

layer with the initial stroke used to classify the bone structures. Figures 6 (b)-(d) show visualizations of different combinations of the generated component functions. Figure 7 shows another application, where we have extracted the layers shown in Figure 4, which lasted approximately two minutes. External users tried the proposed technique on CT angiography data and gave positive feedback. Especially, the ability to directly mark objects in image space was appreciated. Because the 1D transfer function approach constrains the physicians to distinguish features, which do not overlap in the intensity range, they missed segmentation capabilities. Furthermore, the interactive assignment of the optical properties with the subsequent combination of the layers has been considered as effective and fun to do. Until now we did not conduct a complete user study verifying the usability of our approach.

6. Conclusions and future work

In this paper we have proposed a direct approach for specifying transfer functions for volumetric data. We have demonstrated how to design 1D transfer functions without interacting in the histogram domain. Our approach allows the user to identify features directly within the volume rendering, which leads to a more intuitive specification. With the proposed user interface, the user is able to select the optical properties for each feature separately, and it becomes easy to undo transfer function changes, since the user may decide which of the selected features should be integrated into the final transfer function. By using these concepts, we were able to design 1D transfer functions for different data sets.

In the future several extensions could be investigated. Currently, only features which can be classified by using 1D transfer functions can be distinguished. Consequently, supporting the specification of higher dimensional transfer functions or the use of the stroke-based technique for segmentation would be valuable extensions. Furthermore, in analogy to the image processing layer metaphor, it should be possible to choose between different more sophisticated combination techniques when composing the final transfer function. Investigating how the proposed analysis technique responds to different degrees of noise is also an open issue.

Acknowledgements

We would like to thank the reviewers for their comments, which greatly helped to improve the paper. This work was partly supported by grants from the Deutsche Forschungsgemeinschaft (DFG), SFB 656 MoBil Münster, Germany (projects Z1, PM5). The presented concepts have been integrated into the Voreen volume rendering engine (www.voreen.org).

References

[CKLG98] CASTRO S., KÖNIG A., LÖFFELMANN H., GRÖLLER M. E.: *Transfer Function Specification for the Visualization of Medical Data*. Tech. Rep. TR-186-2-98-12, 1998.

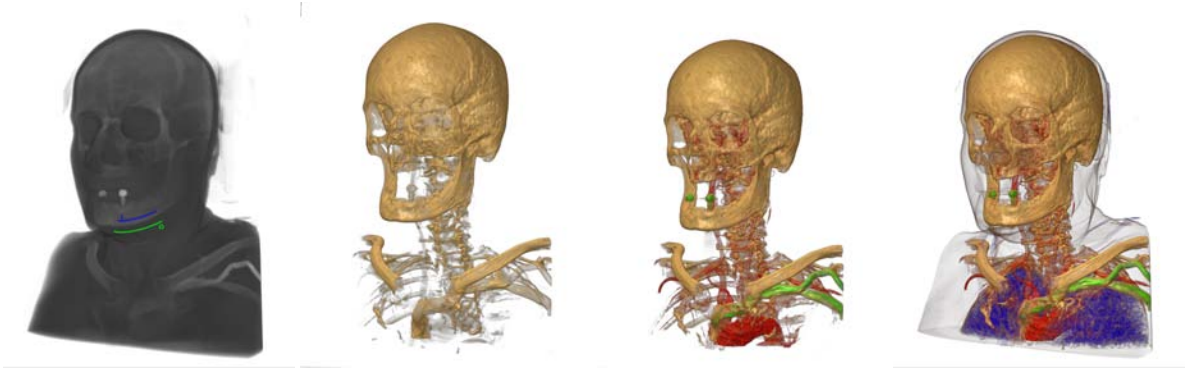
[CSSM06] CHEN H.-L. J., SAMAVATI F. F., SOUSA M. C., MITCHELL J. R.: Sketch-based Volumetric Seeded Region Growing. In *Proceedings of Eurographics Workshop on Sketch-Based Interfaces and Modeling* (2006), Eurographics Association, pp. 123–129.

[DCH88] DREBIN R. A., CARPENTER L., HANRAHAN P.: Volume rendering. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques* (1988), pp. 65–74.

[HM03] HUANG R., MA K.-L.: Rgvis: Region growing based techniques for volume visualization. In *PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications* (2003), pp. 355–363.

[KD98] KINDLMANN G., DURKIN J. W.: Semi-automatic generation of transfer functions for direct volume rendering. In *VVS '98: Proceedings of the 1998 IEEE symposium on Volume visualization* (New York, NY, USA, 1998), ACM, pp. 79–86.

© The Eurographics Association 2008.



(a) (b) (c) (d)

Figure 7: *A 1D transfer function has been assigned to a CT scan of a male's upper body. By drawing strokes (a), we were able to extract and combine the different component functions (see Figure 4) used for rendering (b-d).*

[KG01] KÖNIG A., GRÖLLER E.: Mastering transfer function specification by using volumepro technology. In *Proceedings of the 17th Spring Conference on Computer Graphics 2001* (2001), pp. 279–286.

[KKH02] KNISS J., KINDLMANN G., HANSEN C.: Multidimensional transfer functions for interactive volume rendering. 270–285.

[LSM06] LUM E., SHEARER J., MA K.-L.: Interactive multi-scale exploration for volume classification. In *Proceedings of Pacific Graphics 2006 Conference, also as a special issue of Visual Computer* (2006), pp. 622–630.

[MAB*97] MARKS J., ANDALMAN B., BEARDSLEY P. A., FREEMAN W., GIBSON S., HODGINS J., KANG T., MIRTICH B., PFISTER H., RUML W., RYALL K., SEIMS J., SHIEBER S.: Design galleries: a general approach to setting parameters for computer graphics and animation. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 389–400.

[ONI05] OWADA S., NIELSEN F., IGARASHI T.: Volume Catcher. In *Proceedings of Symposium on Interactive 3D Graphics and Games* (2005), ACM, pp. 111–116.

[PLB*01] PFISTER H., LORENSEN B., BAJAJ C., KINDLMANN G., SCHROEDER W., AVILA L. S., MARTIN K., MACHIRAJU R., LEE J.: The transfer function bake-off. *IEEE Comput. Graph. Appl. 21*, 3 (2001), 16–22.

[RBG07] RAUTEK P., BRUCKNER S., GROLLER E.: Semantic layers for illustrative volume rendering. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1336–1343.

[RSKK06] REZK-SALAMA C., KELLER M., KOHLMANN P.: High-Level User Interfaces for Transfer Function Design with Semantics. *IEEE Trans. on Visualization and Computer Graphics (Proc. IEEE Visualization) 11*, 5 (2006), 1021–1028.

[SBSG06] SEREDA P., BARTROLI A. V., SERLIE I. W. O., GERRITSEN F. A.: Visualization of boundaries in volumetric data sets using lh histograms. *IEEE Transactions on Visualization and Computer Graphics 12*, 2 (2006), 208–218.

[SHN03] SHERBONDY A., HOUSTON M., NAPEL S.: Fast volume segmentation with simultaneous visualization using programmable graphics hardware. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)* (Washington, DC, USA, 2003), IEEE Computer Society, p. 23.

[TLM03] TZENG F.-Y., LUM E. B., MA K.-L.: A novel interface for higher-dimensional classification of volume data. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)* (Washington, DC, USA, 2003), IEEE Computer Society, p. 66.

[TLM05] TZENG F.-Y., LUM E. B., MA K.-L.: An intelligent system approach to higher-dimensional classification of volume data. *IEEE Transactions on Visualization and Computer Graphics 11*, 3 (2005), 273–284.

[WQ07] WU Y., QU H.: Interactive transfer function design based on editing direct volume rendered images. *IEEE Transactions on Visualization and Computer Graphics 13*, 5 (2007), 1027–1040.

© The Eurographics Association 2008.