

User-Centric Transfer Function Specification in Augmented Reality

Bernhard Reitinger¹ Christopher Zach² Alexander Bornik¹ Reinhard Beichel¹

¹Institute for Computer Graphics and Vision
Graz University of Technology
Inffeldgasse 16/II
A-8010 Graz, Austria

²VRVis Research Center
Inffeldgasse 16/II
A-8010 Graz, Austria

{breiting,zach,bornik,beichel}@icg.tu-graz.ac.at

ABSTRACT

The quality of a 3D volume visualization heavily depends on a representative transfer function which is responsible for mapping the original density values to color and opacity. Finding a suitable transfer function is often a tedious task if done manually in a trial-and-error fashion by specifying piecewise linear functions for each color and opacity channel. Contrary, image-based models exploring features like gradient magnitude, histogram, or edge detection do not consider much user interaction as performed almost autonomously. Hence, we propose a new paradigm which integrates the user into the transfer function specification process. This allows an intuitive specification within an Augmented Reality environment by providing different predefined shape functions which can easily be adjusted. Moreover, a new approach is introduced which utilizes spatial information as an additional transfer function. This opens a completely new way of exploration in volume visualization.

Keywords

transfer function, spatial mapping, volume visualization, augmented reality

1 INTRODUCTION

Direct volume rendering (DVR) has established to an accepted visualization method for sampled, computed, or modeled 3D data in recent years. Especially in the medical field, where modalities like computed tomography (CT) or magnetic resonance (MR) generate these datasets, DVR reveals insight to the user which is not or only restricted possible with surface based models. Different software-based visualization methods exist for DVR (see [Lacro94, Malzb93]), however, hardware-based methods have gained much attention because of increasing GPU technologies.

Today's graphics cards often come with at least 128 MB of memory which is enough for reasonable medical datasets. In addition, new hardware features like *texture shaders* or *register combiners* allow fast GPU processing. Recent DVR techniques make use of this hardware functionality providing interactive frame-rates.

Apart from the rendering itself, transfer function specification - a.k.a. intensity classification - plays an important role in this visualization discipline as they specify the mapping between the sampled data and the final color and opacity values. In contrast to thresholding where only one value specifies the visualization output, transfer functions define the color and opacity of one voxel by one-, or even multi-dimensional functions. This allows the user to point out different features which cannot be visualized using indirect rendering techniques (e.g. iso-surface rendering) using one threshold value.

The only problem is that finding a good transfer function is often discouraging, as being a hard and tedious task. Hence, users are often overcharged. This is because the mapping is specified by drawing piecewise linear functions. Contrary, image-based methods which include information like gra-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Journal of WSCG, Vol.12, No.1-3., ISSN 1213-6972
WSCG'2004, February 2-6, 2004, Plzen, Czech Republic.
Copyright UNION Agency - Science Press

dient magnitude, histogram, or others allow too little user interaction.

This paper is concerned about these problems and suggests a method for defining user-centric transfer functions in a quick and easy way. In this paper, we focus on opacity mapping functions. By taking advantage of an Augmented Reality (AR) user interface, the user is able to apply and adjust different kinds of predefined transfer functions easily. Additionally, an approach is presented which integrates spatial information as a new transfer function. This means in particular, that different regions of interest can be specified which in turn act as an additional transfer function. Consequently, different parts of the volume can be pointed out, while unimportant regions are invisible.

The paper is organized as follows: Section 2 discusses related work for different transfer function specification methods and volume rendering techniques. Section 3 outlines volume rendering within Augmented Reality environments. Section 4 presents different simple predefined transfer functions which target on intensity values of the volume dataset. Additionally the novel spatial-dependent transfer function model is explained. Before showing some results in Section 6, the use within an AR environment is shown. Finally, conclusions and future work are presented.

2 RELATED WORK

2.1 Transfer Functions

Although direct volume rendering has been and still is an active research field in the last years, only few groups have focused on the task of making transfer functions specification easy and intuitive to the user. He *et al.* [He96] generate transfer functions by stochastic search techniques. This search is treated as a parameter optimization problem and addresses different stochastic techniques. This method has been integrated into the VolVis package developed at SUNY Stony Brook [Avila94]. By using a thumbnail gallery of pre-rendered images, the user can select the best transfer function. Another gallery approach is presented in [König01] where small thumbnails are generated and arranged according to their relationship with spaces of data values, color, and opacity. A quite recent work of Botha and Post presents a fast method for finding the suitable transfer function by an immediate fast visual feedback model [Botha02]. The main motivation for

this work is software-based rendering where an update of the transfer function is not an interactive task.

An image-based method is presented by Fang *et al.* [Fang98] where they integrate different 3D image processing tools into the volume visualization pipeline to facilitate the search for an image-based transfer function. Different operations like image smoothing or different sharpening methods are included. User interaction is not demanded except parameter tuning. A semi-automatic generation of transfer functions is presented in [Kindl98]. This approach is also image-based and includes histogram based methods. User interaction is also restricted to parameter setting.

Aside from all these methods, Kniss *et al.* have presented a method for using multi-dimensional transfer functions based on data value, gradient magnitude, and the second directional derivative [Kniss02]. By applying this new technique different material boundaries are easy to generate. However, the user interface gets more complex by using multi-dimensional transfer functions. Direct manipulation widgets are introduced which allow the adjustment of each dimension.

In [Pfist01], an evaluation of four different transfer function specification methods is presented. One method is “trial-and-error”, two methods are data-centric, and one method is image-centric. In the conclusion of this evaluation it turned out that it doesn’t matter if the technique is automatic, semi-automatic or manual unless it is fast and simple for the user. This conclusion - **to be fast and simple** - has motivated us to think about a new way for transfer function specification.

2.2 Hardware Volume Rendering

As transfer functions are always combined with direct volume rendering, we outline some recent work and techniques very briefly for a better understanding.

Apart from various software volume rendering techniques, different hardware-based methods have been presented in recent work. By using texture memory, the data volume is sampled, classified, and rendered to a proxy geometry exclusively on the graphics card. In that case, transfer functions are realized as a lookup table which is itself a 2D texture¹.

¹Note: *dependent textures* in OpenGL are realized through 2D textures, although they contain 1D data in this case.

In hardware volume rendering, two different techniques are present. The first method is an object-aligned approach and uses 2D textures along major axis of the data [Rezk-00]. The main disadvantage is that three copies of the volume must be kept in memory, each for one axis. The second method is called view-aligned and uses a 3D texture buffer to keep the volume data [Engel02]. The typical proxy geometry are parallel quads. However, spherical shells are also very popular to eliminate the artifacts caused by perspective projection [LaMar99].

3 AUGMENTED REALITY BASED VOLUME RENDERING

Volume visualization is often integrated in some desktop-based applications. This is mainly because new developed algorithms are easier and faster to test on the same machine where the algorithms are implemented, namely on a common desktop machine. However, for some applications it is desirable to integrate volume visualization into a 3D Virtual or Augmented Reality environment with completely different user input devices. In our case, volume visualization is an integrated part of the liver surgery planning system (LSPS) which is an on-going research project and enables physicians to visualize and refine segmented liver datasets, as well as to simulate and evaluate different resection plans within an AR environment [Borni03]. LSPS uses the *Studierstube* as an AR library which provides interaction with a pencil and a panel [Schma96].

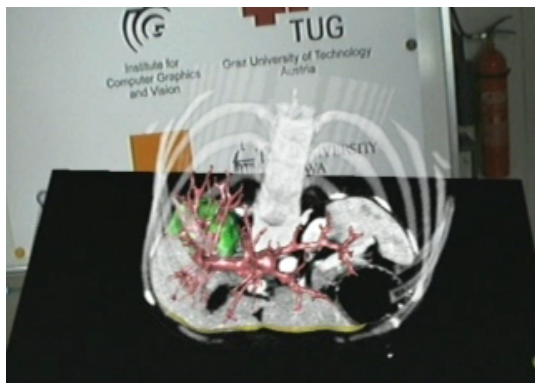


Figure 1: Volume rendering in LSPS showing additional surface rendering and CT context information.

Figure 1 shows our AR volume visualization including surface rendering of a segmented vessel tree and a green colored tumor. Additionally, CT context information can be displayed on the backside of the panel reflecting the original volume dataset. The first implemented transfer function specification method uses the panel to draw piecewise linear functions for the specification (see Figure 2). For each channel a mapping function could be drawn free-hand in a trial-and-error fashion. However, this first approach leads to several difficulties. Firstly, drawing lines freehand on the panel by using the pencil turned out to be very difficult and not feasible for physicians. Secondly, finding a good transfer function in this trial-and-error fashion is a very hard task, especially if physicians are interested in specific parts of the volume. Hence, we have tried to find an alternative way which accounts for 3D interaction and is not possible on desktop-based systems.

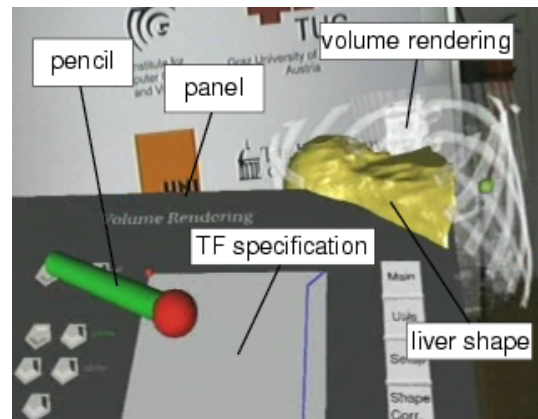


Figure 2: Transfer function specification with piecewise linear functions.

4 TRANSFER FUNCTIONS

As volume visualization as well as surface visualization are integrated into the LSPS, the former is mainly used to get context information of the original unsegmented dataset. Two different aspects are taken into account for volume visualization. Firstly, different predefined intensity-based transfer functions should be provided which allow an efficient way of visualizing volume parts of similar intensity values. Secondly, we introduce a new spatial-dependent transfer function which allows to highlight volumes of interest. By setting an arbitrary point-of-interest within the volume, a distance map is calculated which in turn influences

the opacity of neighboring voxels. To our knowledge, this kind of transfer function specification is a unique approach up to now. For the rest of this section, these two methods are explained and the technical details are described.

4.1 Intensity-Based Transfer Functions

The idea behind our user-centric intensity-based transfer function specification is to provide predefined shapes which allow voxel highlighting of similar intensity values. This is because in the medical field, physicians often like to see other (not segmented but almost homogeneous) organs as context information to segmented surface-based meshes. The predefined functions are controlled by a *peak* value which represents a demanded intensity threshold. Based on this *peak* value, the following shape functions can be applied up to now which have been proven to be useful within our environment:

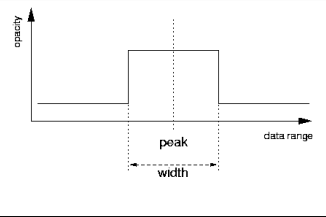
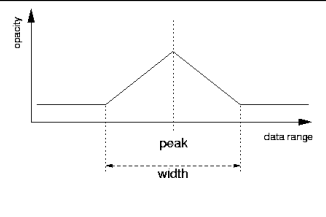
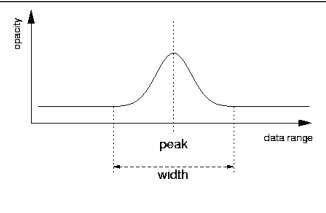
<p>Box shape</p>	
<p>Tent shape</p>	
<p>Gaussian shape</p>	

Table 1: Different shapes for predefined transfer functions.

Each shape uses the *peak* value as the position for the function’s amplitude which can be in range [0..255]. Therefore, the *peak* must be representative value. As it would be hard to specify the *peak* value by hand, we have designed a method which allows an easy *peak* selection. As already pointed out, original CT intensity values are displayed on the backside of the panel. By moving the panel through the volume, an arbitrary *peak* value can

be selected using the pencil (see Figure 9). As the intensity value of one voxel is not representative, the *peak* value also considers neighboring voxels. This is done by calculating the median value of the intensity values for the selected voxel and of its 26-connected voxel intensity values. If more than one *peak* value is selected, the maximum of all function values is used as opacity value in overlapping areas (see Figure 3)

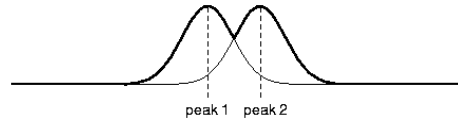


Figure 3: Example of two overlapped shape functions.

4.1.1 Texture-Based Transfer Function Lookup

As we use 3D texture-based volume rendering, the transfer function is also stored on the graphics card memory. Therefore, low-level *texture shaders* are used in order to implement the transfer function as a lookup table. By using different texture units (ARB_multitexture OpenGL extension) the intensity volume buffer and the lookup table are stored in two consecutive units. A shader program allows a dependent texture lookup. This means in particular, that the result of the previous texture lookup is used as texture coordinates for the subsequent texture fetch. We have used the *dependent alpha-red* shader program, where red and alpha values from the previous texture define 2D texture coordinates for the dependent texture fetch. Due to this mechanism, transfer function replacement is very fast. If the user selects different *peak* values, only the lookup table must be updated.

4.2 Spatial-based Transfer Function

Additionally to common intensity-based transfer functions, we introduce a new mapping possibility which has not been explored in volume rendering so far. This idea has been born while implementing the AR *peak* selection method. Apart from the intensity value of the selected voxel, we also consider the current pen position as a reference point within the volume. According to this reference point, a distance map is generated which reflects the Euclidean distance from each voxel to this reference point. If multiple reference points ($r_1..r_n$) are specified the distance value *dist* of voxel *v* is

calculated according to the following formula:

$$dist_v = \min(\|r_1 - pos_v\|, \|r_2 - pos_v\|, \dots, \|r_n - pos_v\|)$$

This information is stored as an additional 3D texture in a separate unit. Figure 4 shows a sample color coded distance map where two different reference points are specified.

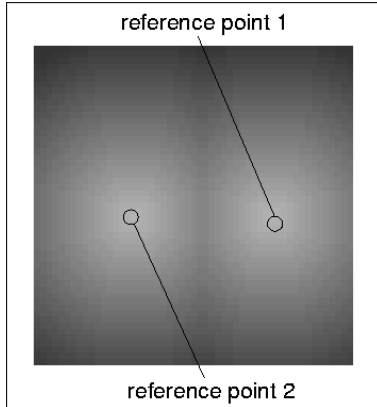


Figure 4: Sample color coded distance map showing two different reference points.

A distance function (spatial-based transfer function) can be assigned which controls the impact of the distance map on the volume buffer. Similar to the intensity-based transfer function, the spatial-based transfer is also implemented as a *dependent texture* lookup table.

A various number of shapes can be applied as distance function. However during experiments, two different shapes turned out to be useful. The first one is a linear function which is controlled by a certain threshold. Opacity values farther away from the reference point are getting darker. The second one is a step function where alpha values beyond a certain threshold are culled off (see Figure 5).

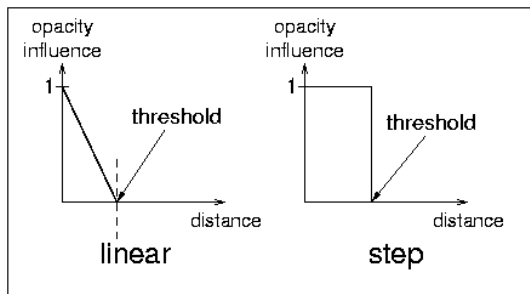


Figure 5: Two feasible distance functions.

4.3 Final Transfer Function Pipeline

The problem now is to combine these two presented one-dimensional transfer functions on the graphics card. Firstly, the intensity buffer is loaded as a 3D texture into the first texture unit. A selected intensity-based transfer function is stored as a 2D dependent texture into the second unit. According to the reference point position, a distance map is calculated and also uploaded to the graphics texture memory as a 3D texture on unit 3. The selected distance function resides as a 2D texture in unit 4. After all texture fetches, we have two 2D and two 3D textures in the graphics memory. Finally, the alpha value of unit 1 must be multiplied with the output of unit 4 by using *register combiners* (see Figure 6).

5 REVISED USER INTERACTION

After explaining the main components of our methods, all parts are joined together to show their use within an AR environment. As one of our goals is to allow an easy and fast transfer function specification, we have redesigned the existing panel which is now reduced to few simple widgets (see Figure 8). Two different sliders are necessary to parameterize the amplitude and the peak value for the intensity-based transfer function. An additional slider controls the threshold value for the distance function. The adjusted functions are displayed on the panel.

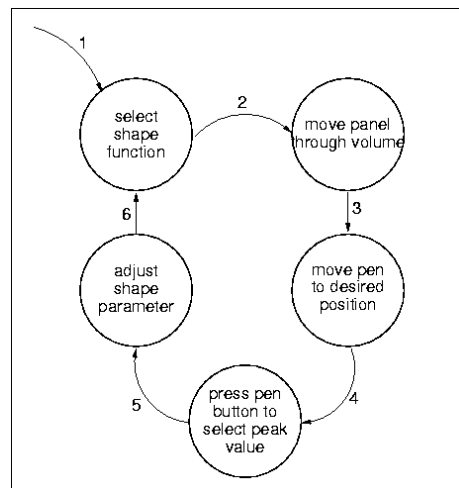


Figure 7: User interaction for transfer function specification.

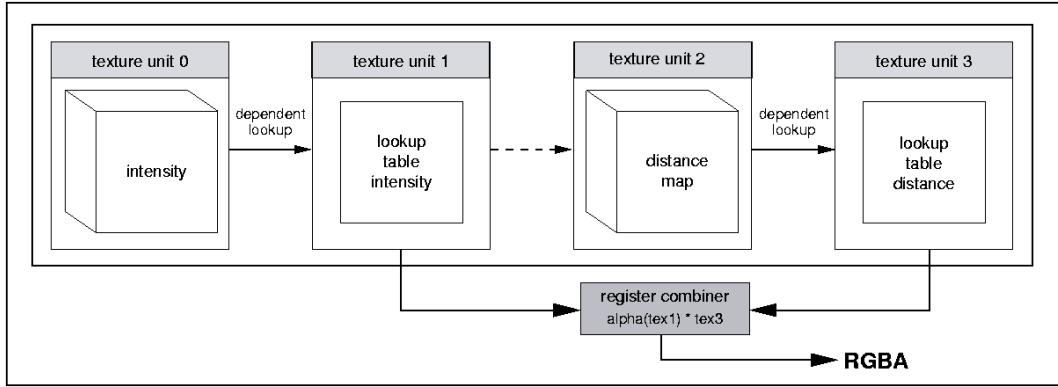


Figure 6: Multi-texturing using texture shader and register combiners.

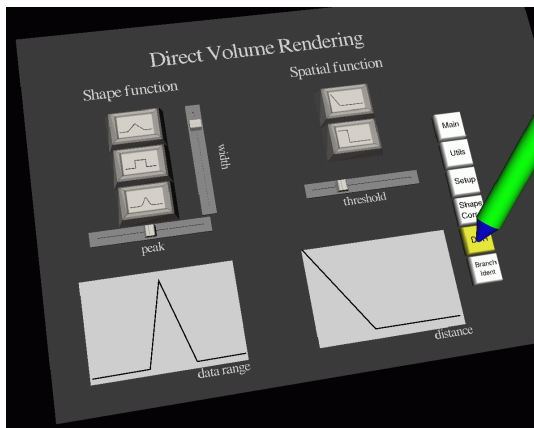


Figure 8: New panel design.

Figure 7 illustrates the specific tasks which are necessary to apply intensity-based as well as spatial-dependent transfer functions on direct volume rendering. Firstly, the user has to select the desired shape function which should be used for each domain. Then, the panel must be moved through the CT volume where intensity values are projected on the panel according to the panel's orientation. If the desired position is found, the intensity value (*peak*) is selected by moving the pen towards the panel and by pressing the button on the pen (see Figure 9). The previous selected shape functions for both domains are then applied immediately. Moreover, additional *peak* values can be defined in the same manner.

6 RESULTS

The methods presented in this paper are integrated into our liver surgery planning system.

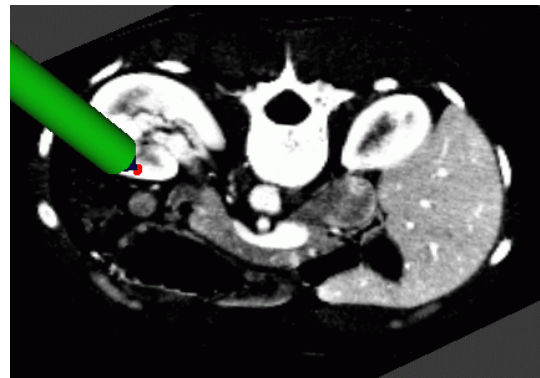


Figure 9: Selecting the *peak* value by using the context information on the panel.

By using a pencil and a panel as input devices, intensity-based and spatial-dependent transfer functions can easily and interactively be defined within the AR environment. A nVidia Quadro4 900 XGL is used which supports quad-buffered stereo and 3D textures. This allows to use direct volume rendering at interactive frame-rates.

The AR pictures in Figure 1, 1, 8, and 9 are taken by using a tracked camera which captures the real and the virtual scene at the same time. The picture plate on the last page (Figures 10, 11, 12) presents some screenshots of different volume rendered datasets where all presented methods have been applied.

7 CONCLUSIONS

This paper has presented a new user-centric Augmented Reality approach for specifying predefined as well as spatial-dependent transfer functions. By using a pencil, interaction with the volume is kept

very easy and intuitive. Parameters of predefined shape functions can be modified by using sliders on the panel. Compared to our previous panel interaction, transfer function specification is much simpler now. However, an evaluation must be initiated in order to validate our approach.

Additionally, a new concept of taking spatial information into account has also been discussed. This allows to outline volumes of interest not by modifying the original dataset but by adding an additional transfer function.

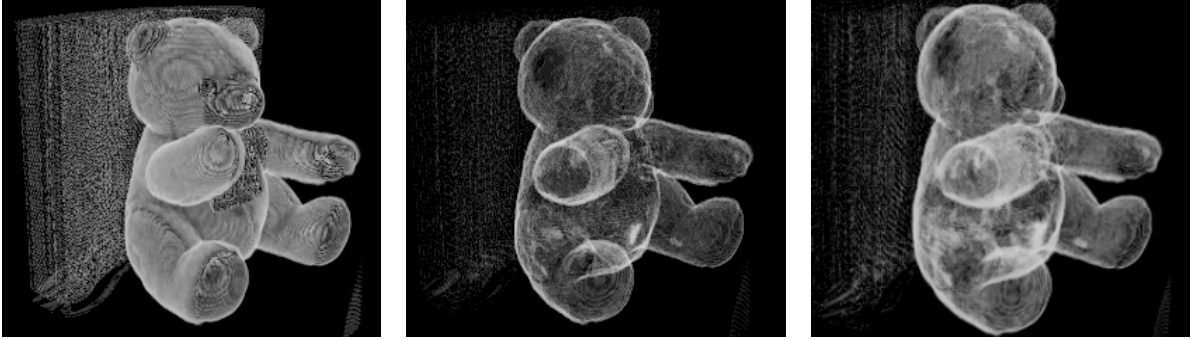
For future work, we plan to include color support into this framework. Additionally, improvements for calculating the distance map are planned, where the map will be calculated directly on the GPU using higher-level shading languages.

ACKNOWLEDGMENTS

This work was supported by the Austrian Science Foundation (FWF) under grant P14897. Furthermore, we would like to thank Gustavo Fernandez for his comments and suggestions.

References

- [Avila94] R. Avila, T. He, L. Hong, A. Kaufman, H.-P. Pfister, C. Silva, L. Sobierajski, and S. Wang. VolVis: a diversified system for volume visualization research and development. In *Proc. of Visualization '94*, pages 31–38, 1994.
- [Borni03] A. Bornik, R. Beichel, B. Reitering, G. Gotschuli, E. Sorantin, F. Leberl, and M. Sonka. Computer aided liver surgery planning: An augmented reality approach. In R.L. Galloway, editor, *Medical Imaging 2003, Proceedings of SPIE*, volume 5029. SPIE Press, May 2003.
- [Botha02] C.P. Botha and F.H. Post. New technique for transfer function specification in direct volume rendering using real-time visual feedback. In *Proc. of the SPIE Int. Symposium on Medical Imaging*, volume 4681, 2002.
- [Engel02] K. Engel and T. Ertl. Interactive high-quality volume rendering with flexible consumer graphics hardware. In Eurographics, editor, *STAR – State of the Art Report*. Eurographics '02, 2002.
- [Fang98] S. Fang, T. Biddlecome, and M. Tuceryan. Image-based transfer function design for data exploration in volume visualization. In *Proc. IEEE Visualization'98*, pages 319–326. IEEE Computer Society, 1998.
- [He96] T. He, L. Hong, A. Kaufman, and H.-P. Pfister. Generation of transfer functions with stochastic search techniques. In *Proc. IEEE Visualization 1996*, pages 227–234, 1996.
- [Kindl98] G. Kindlmann and J.W. Durkin. Semi-automatic generation of transfer functions for direct volume rendering. In *IEEE Symposium on Volume Visualization*, pages 79–86, 1998.
- [Kniss02] J. Kniss, G. Kindlmann, and C. Hansen. Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):270–285, 2002.
- [König01] A. König and E. Gröller. Mastering transfer function specification by using volume projection technology. In *Proc. of Spring Conference on Computer Graphics (SCCG) 2001*, pages 279–286, 2001.
- [Lacro94] P. Lacroute and M. Levoy. Fast volume rendering using a shear-warp factorization of the viewing transformation. In *SIGGRAPH '94*, pages 451–458, 1994.
- [LaMar99] E. LaMar, B. Harmann, and K.I. Joy. Multi-resolution techniques for interactive texture-based volume visualization. In *Proc. of IEEE Visualization'99*, pages 355–361, 1999.
- [Malzb93] T. Malzbender. Fourier volume rendering. *ACM Transactions on Graphics*, 12(3), July 1993.
- [Pfist01] H.-P. Pfister, B. Lorensen, C. Bajaj, G. Kindlmann, W. Schroeder, L.S. Avila, K.M. Raghu, R. Machiraju, and J. Lee. The transfer function bake-off. *IEEE Computer Graphics and Applications*, 21(3):16–22, 2001.
- [Rezk-00] C. Rezk-Salama, K. Engel, M. Bauer, G. Greiner, and T. Ertl. Interactive volume on standard PC graphics hardware using multi-textures and multi-stage rasterization. In *Proceedings 2000 SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 109–118, Interlaken, Switzerland, August 2000.
- [Schma96] Dieter Schmalstieg, Anton L. Fuhrmann, Zolt Szalavari, and M. Gervautz. Studierstube – an environment for collaboration in augmented reality. In *Proceedings of Collaborative Virtual Environments*, pages 19–20, 1996.

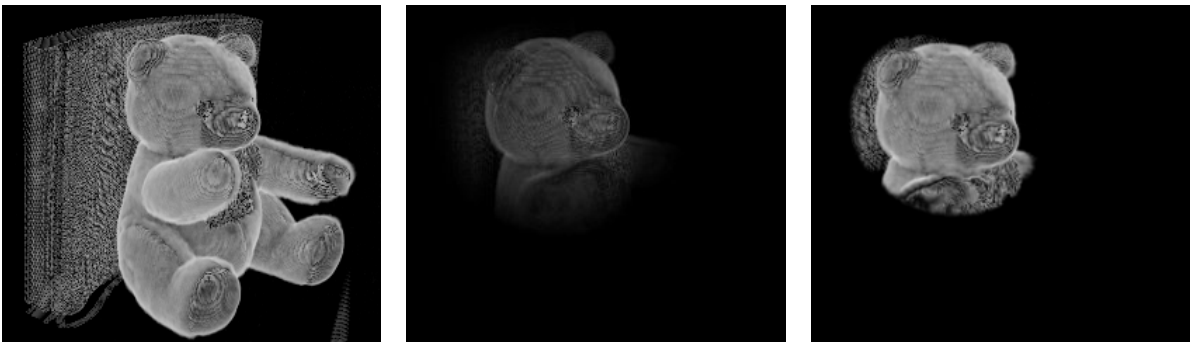


(a)

(b)

(c)

Figure 10: **Teddy dataset** (128x128x62): intensity-based (a) tent function, *peak*: 70, *width*: 78, *amplitude*: 255, (b) box function, *peak*: 39, *width*: 12, *amplitude*: 32 (c) gauss function, *peak*: 43, *width*: 0.029, *amplitude*: 64.



(a)

(b)

(c)

Figure 11: **Teddy dataset** (128x128x62): spatial-dependent (a) tent function, *peak*: 70, *width*: 84, *amplitude*: 255, (b) linear distance function, *peak*: 70, *width*: 84, *amplitude*: 255, *threshold*: 30, (c) step distance function, *peak*: 70, *width*: 84, *amplitude*: 255, *threshold*: 20.



(a)

(b)

(c)

Figure 12: **Thorax CT dataset** (256x256x96): (a) gauss function, *peak*: 212, *width*: 0.002, *amplitude*: 32, (b) step distance function to point out left kidney, *peak*: 70, *width*: 84, *amplitude*: 255, *threshold*: 15, (c) 2 step distance functions to point out both kidneys, *peak*: 70, *width*: 84, *amplitude*: 255, *threshold*: 15.