Two-Level Interaction Approach for Transfer Function Specification

João Luis Prauchner   Carla M.D.S. Freitas   João L.D. Comba

Instituto de Informática, Universidade Federal do Rio Grande do Sul
{jlprauchner, carla, comba}@inf.ufrgs.br

Abstract

Direct volume rendering techniques are used to visualize and explore large scalar volumes. Transfer functions (TFs) that assign opacity and color to scalar values are very important to display volume features, but their specification is not trivial or intuitive. This work presents an interactive, semi-automatic tool to assist the user in the generation of opacity and color TFs. We use the histogram approach proposed by Kindlmann and Durkin [9] to reduce the scope of candidate TFs presented to the user following the Design Galleries method [18]. The combination of these two solutions leads to a single interactive tool that allows the user to deal with different aspects of TF specification.

1. Introduction

Volume rendering techniques are well described in the literature [7, 16, 20, 27]. Methods based on isosurface extraction (see, for example, [17]) render a 3D mesh constructed from the volume using the standard graphics pipeline. On the other hand, direct volume rendering techniques compose data information without building an intermediate geometry [15]. Recent advances in graphics hardware allowed real time high-quality volume visualization [4, 14] for both approaches.

An important aspect for extracting iso-surfaces as well as to directly render the interior of volumes is the determination of which voxel values correspond to the region of interest. In direct volume rendering this is a mapping from the original values associated with each voxel to visual (image) attributes, such as color and opacity. This mapping is often represented by transfer functions, which are normally pre-computed and stored in lookup tables.

Transfer functions (TFs) are particularly important to obtain correct (and good quality) images because they provide a way to classify the voxels. However, the task of specifying TFs in order to convey the required information is not trivial and has been widely discussed [8, 23]. Often, opacity levels are defined by editing control points in a line graph, which represents the mapping from voxel values to opacity levels. The problem with this approach is that subtle changes in the TF result in drastic changes in the rendered image. This leads the user to an unintuitive trial and error task, which is time-consuming.

There are also several automatic and semi-automatic techniques for specifying TFs [1, 5, 9, 10, 12, 13, 18], and due to their complexity, interactive interfaces to assist the user in this task have been developed [11, 25].

We present a method that combines the techniques proposed by Marks et al. [18] and Kindlmann and Durkin [9] into a single, interactive tool. There are two levels of interaction in our method. The first one refines a set of initial TFs rendered as 3D thumbnails, obtained from a previously calculated 3D histogram. Then, the user chooses the most appealing thumbnail, and the volume is rendered based on the respective TF. At this moment, the second level of interaction allows further refinement of the TF.

The paper is organized as follows. The next section discusses relevant previous work. Sections 3 and 4 explain our method and the implementation, while Section 5 shows results obtained with the application of our method to sample datasets. The paper ends with conclusions and future work in Section 6.

2. Previous Work

The task of specifying TFs has also received considerable attention in the literature. There are good surveys describing methods for TF specification and the trade-offs between them [8, 23], so we restrict our comments to methods that are most relevant to our work.

Existing techniques can be classified as data-driven or image-driven methods [8]. Basically, data-driven methods extract information from the scalar values which comprise the volume [1, 9, 12, 21]. This information guides the user in specifying TFs, or constrains the large parameter space involved in their generation. Image-based techniques use volume rendered images to guide navigation in the TF space [18, 25]. Thus, the user chooses the most appealing rendered images instead of specifying the TF.

Traditional approaches [13] let the user manually specify the transfer function by editing control points in a graphic plot. If the user does not have much knowledge *a priori* about the dataset, this task may become a trial-and-error procedure [23]. Bajaj et al. [1] presented a data-driven technique aimed at extracting isosurfaces of interest from unstructured volumetric grids. Metrics such as mean gradient magnitude, surface area, and volume are extracted from the grid and shown in the same interface where the isovalue is specified. Marks et al. [18] proposed Design Galleries, an image-driven method devoted to the specification of parameters for rendering synthetic 3D scenes, but including the specification of TFs for direct volume rendering. This approach consists of randomly generating a large set of TFs and rendering them as 2D thumbnails, which are arranged according to their similarities in order to facilitate browsing. The user then interacts with the tool by selecting and visualizing one of these images.

Kindlman and Durkin [9] presented their data-driven, semi-automatic technique based on observations from Levoy [15] about using gradient magnitude to enhance material boundaries in the volume. In their semi-automatic method, a 3D histogram is extracted from the volume. This histogram records the relationship between a voxel and its first and second derivatives along the gradient direction. From the histogram, distance maps are extracted, indicating the distance from a given voxel value to the nearest boundary in the volume. The user then specifies a boundary emphasis function, which assigns opacity to voxels according to the distance maps. More recently, Kniss et al. [11] extended this work by developing an interface for specifying multidimensional TFs through the manipulation of graphic widgets. These widgets allow the assignment of color and opacity in both transfer function (with a graphic plot) and space domains (with a pencil that points regions of interest in the volume). Pekar et al. [21] presented an improved method of extracting isocontour information from volumes. A cumulative Laplacian-weighted gray value histogram is calculated in a single pass on the volume, the global maximum of the histogram curve being the most dominant material transition. This information is used to set up the opacity TF in a VTk [24] application.

Multidimensional TFs have additional specification complexity because the parameter space grows exponentially with the number of dimensions. However, with a larger number of dimensions, one can more accurately classify different materials that share similar voxel values. Medical datasets, such as CT and MRI scans, often present these situations. Multidimensional TFs also suffer from large storage requirements and may become unusable at higher dimensions.

In order to address this challenge, Kniss et al. [12] proposed the use of Gaussian primitives to represent TFs instead of lookup tables. These primitives are explicitly evaluated using programmable graphics hardware. Besides their compact representation form, Gaussian TFs also allow more accurate evaluation of the volume rendering integral on the GPU. This produces good quality images with fewer slices. Recently, Tzeng et al. [25] developed another image-driven technique, focusing on volume segmentation tasks in biomedical datasets. In the proposed interface, the user interacts by painting regions of interest in the volume slices. A neural network implemented as a fragment program in the GPU is trained to assign opacity to these regions in order to enhance them, using information such as gradient magnitude, voxel spatial position and neighborhood. The interface also allows the removal of non-interesting materials by painting on sample slices.

Our method combines features from previous work [9, 18] into a general-purpose tool that overcomes some of their limitations.

3. Two-Level Interaction TF Specification

In traditional volume rendering interfaces, the user is required to have a certain degree of knowledge about the dataset being visualized to obtain a good TF. Moreover, each new dataset presents new challenges and the user might spend a large period of time in defining acceptable mappings [6]. On the other hand, methods that exclude the user from the process of TF specification are often not adequate [22] because they might miss important details only noticeable to a human observer. In our approach, users do not need to have a priori knowledge about the dataset, since the initial TFs are automatically generated. But if that understanding exists, they can take advantage of it by refining the initial TFs or regenerating the whole set or a subset of TFs.

Among the several techniques described in the literature, the work of Kindlmann and Durkin [9] and the Design Galleries method [18] are particularly interesting for our purposes of implementing a tool for TF specification. More specifically, rendering several randomly generated thumbnails may allow the user to gain some insight about the data immediately, before interaction. A problem that affects this approach is that usually volume data presents some material of interest completely surrounded by other (less important) materials. If opacity is assigned to uninteresting voxels, they may occlude important inner features, thus affecting the desired visualization. In order to solve this problem, one can consider the regions of interest as the boundary between different homogeneous regions [9]. Another issue regarding the Design Galleries method is its excessive automation, since the TFs are randomly generated and dispersed to enhance variety. We are interested in reducing the parameter space of TFs to a more reasonable one.

We propose an extension of the two mentioned techniques, combining some aspects of them to develop a two-level interactive tool aimed at TF specification. The overall process of classification and visualization in our technique is shown in Figure 1 and explained in the next sections.

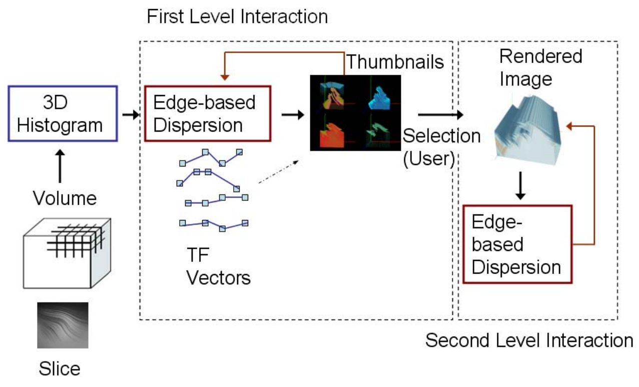

Figure 1. The interactive TF specification method. The edge-based dispersion is used for generating initial TFs resulting different thumbnails, and also during the second-level, producing different volumetric images.

3.1. Preparing Data for Initial TFs

Since one of our goals is to prevent the user from the trial and error approach, our method starts with the automatic generation of initial TFs. In the Design Galleries method [20], this is performed by disturbing input vectors, which represent the control points of opacity TFs. After rendering the TF vectors as thumbnail images, the Design Galleries method employs a dispersion process in the initial TFs. This process uses neighboring pixels from the rendered images and, based on their similarity, decides if a different TF must be generated in order to provide a better variety of the resulting thumbnails.

Instead of showing the largest number of random possibilities, our method tries to directly find combinations of TFs that enhance the boundaries between different regions. By doing this we try to constrain the large space of possibilities to a more promising one. We calculate a 3D histogram, following [9], recording the relationship between a voxel value and its first and second directional derivatives along the gradient directions in a way that the voxel spatial position is not needed. Indeed, it is observed that the domain of 1-D TFs does not include position, only the voxel value.

Since the first (*f'*) and second (*f''*) derivatives in an image encode the rate in which the values are changing in a given direction, the points of maximum *f'* derivative, as well as zero-crossings of *f''* (along the gradient direction) indicate a change in the scalar values, or a boundary between homogeneous regions. One axis or dimension of the volume histogram is the voxel value. The others are *f'* (or the gradient magnitude) and *f''* derivative, both measured at each voxel in the volume. We used central differences to measure the first derivative (gradient magnitude) and the Laplacian operator to approximate the second derivative, both with 3x3x3 convolution kernels. This calculation is simple to implement yet is accurate enough to detect several different boundary configurations. The resulting histogram records the number of occurrences of a given combination of data value, gradient magnitude and second derivative in the whole dataset.

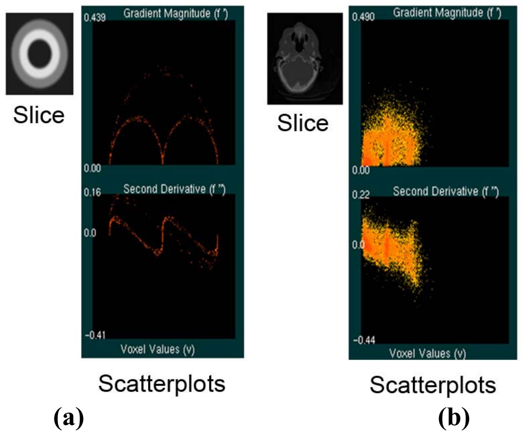

Figure 2. Projections of the 3D histogram: (a) synthetic dataset of a cylinder; (b) UNC CT head. In both examples it is possible to identify boundary regions through the points of maximum gradient magnitude (f') and zero-crossings of f''.

Similar to [11], we show the scatterplot of the 3D histogram, which are projections in the axis of data value versus *f'* and data value versus *f''* (Figure 2). The horizontal axis corresponds to the voxel value range and the vertical axes to the *f'* and *f''* ranges. To enhance few hits, we map the number of occurrences to colors, varying from yellow (few occurrences) to red (several occurrences).

3.2. Initial TFs and First-Level Interaction

As observed in the scatterplots shown in Figure 2, each voxel value often has several different measures of *f'* and *f''*. The average of *f'* and *f''* is computed for each voxel value and weighted by the number of total hits of that data value. These averages serve as score indicators, and tell us if a given value potentially belongs to a transition region. We then order these values, considering decreasing order of *f'* averages (since we are interested in the points of maximum *f'*), and proximity to zero in *f''* (since we are also interested in the zero-crossings of *f''*).

To build the initial TFs, we propose an edge-based dispersion method. A set of 6x6 initial TFs is built through the initialization of 8-entry vectors, as follows. A vector with eight entries (or control points) is filled with the values with the highest scores computed for *f'* (as explained before). Then, five more vectors are generated based on this one: randomly, the method determines the number of non zero-opacity values to be assigned to the vectors; for each non zero-opacity entry, the method randomly selects an opacity value and a color. Each vector is then interpolated to generate a full lookup table, with the same number of entries as different scalar values. The next subset of 6 vectors is also built in the same way by varying opacity and color for the values in the first vector.

The next two subsets are built based on the highest scores of *f''*, and the final two ones based on the highest values of a function of *f'* and *f''*. Thus, each vector represents a different TF that specifies the amount of opacity and color assigned to each entry, which is indexed by voxel value. Although all the vectors have eight positions or control points, fewer positions have also shown effective for most cases.

The initial set of TFs is rendered as 3D thumbnails in real time using programmable graphics hardware. Each thumbnail is a different viewport arranged in the same window, as illustrated in Figure 3.

One of the key issues in TF specification is its dependence on both the application and the dataset being visualized. Some combination of parameters presents better results when applied to some datasets, but not for all of those we tested. These parameters are: (1) the maximum opacity given to a scalar; (2) the number of different control points in a TF vector that actually will be assigned non-zero opacity, and (3) the criteria used to choose the boundary voxels, which can be the highest *f'* values, the zero-crossings of *f''*, or a score obtained by dividing *f''* by *f'* and modulated by the so-called *thickness* of the boundary [18].

Other factor that affects the resulting images is the choice of interpolating (or not) the TF, since this interpolation can assign opacity to uninteresting regions. To generate the initial set of TFs, we randomly pick parameters for each vector, trying to obtain a successful combination for the user needs.

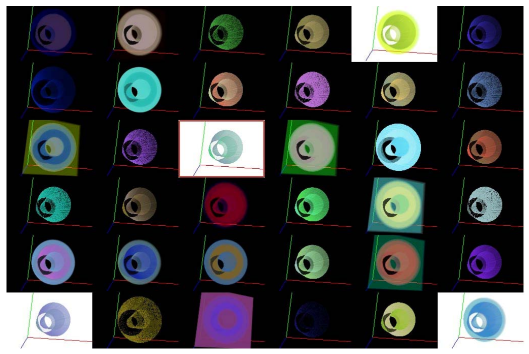
Figure 3. Set of initial TFs for the cylinder dataset rendered as 3D thumbnails. The ones shown in white background were saved and will not change in a new generation of TFs.

After the display of the initial TFs' thumbnails, the user can enter the first-level interaction cycle. The user can zoom in/out and rotate all of thumbnails, and select the most appealing one for a posterior refinement (if necessary). At the second level of interaction, some parameters involved in the automatic generation of the initial TFs can be modified to generate a whole new set of TFs. Moreover, instead of re-generating all the thumbnails, the user can save selected images (shown in white background in Figure 3) thus preserving them from overwriting.

Another interesting feature of our tool is the ability to perform the dispersion with respect to a selected image, which generates new dispersed TF vectors, but maintains characteristics of the chosen image. We do this by modifying the parameter set for the vector which generated that image, creating a slightly different one each iteration.

3.3. Second-Level Interaction

Once the user has selected a thumbnail in the first level of interaction, the correspondent TF is used to render an image from the volume, and the user can enter the second (and optional) level interaction loop. The user can refine the obtained TF by interacting with the dataset, rotating and zooming in/out the image in the same way as with the thumbnails.

The graphic plot of the TF is displayed, together with the control points of the respective generation vector. At this time, the user may choose between going back to the thumbnails, returning to the first-level interaction, or modify the parameter set with immediate feedback in the volumetric image.

Since color can also be defined at this level, we decided to use a color picker because of its ease of use. Figure 4 illustrates this idea. Figure 4a is a single slice of the volume. Figure 4b is shown using the TF associated to a selected thumbnail. The image in Figure 4c was obtained using the same opacity TF, but with different colors chosen in the color picker after selecting the materials of interest. The TF is then updated and the new resulting image is displayed in real time. If the user is not satisfied with the refinement, it is always possible to return to the first-level interaction and search new TFs or undo the changes made.

3.4. Rendering

Texture-based volume rendering [2, 26] is used in our system. Volume data is stored into 3D textures and rendered by displaying a proxy geometry consisting of planes orthogonal to the viewing direction [16, 3]. Since we desired interactive rates, we chose to implement a post-classification scheme [14, 19] as a fragment program executed on the GPU. Each fragment queries a lookup table to fetch the opacity and color values. These values are then composed in the frame buffer and the resulting image is generated.

4. Results and Discussion

Our results were obtained on a computer with a 1GHz processor, 512 MB of RAM and a GeForce FX 5200 graphics card with 128 MB. Images shown in Figures 5-7 were rendered at interactive rates (approximately 20 frames per second). Dataset dimensions varied from 64^3 to 256^3. The interface allows the user to choose the number of proxy planes, from the fastest configuration but with lower quality images (40 planes) to the slowest configuration (2500 planes) but generating higher-quality images.

Figure 5 shows the cylinder dataset (size 64^3). After interacting with the thumbnails shown in Figure 3, the set of thumbnails represent different, modified TFs. Two of them are selected and used to render the two rightmost images.

Figures 6 and 7 present screenshots of user interaction with our tool applied for visualizing well-known datasets, the engine block dataset, and the UNC head CT, respectively (videos are available at [28]). In all examples, the thumbnail interface is displayed on the left, illustrating the first level of interaction. The thumbnails shown in these figures were obtained through selection, dispersion, and re-generation after approximately five user refinement iterations. Notice that despite the variety and difference between these thumbnails, none of them assigned opacity to the outer material (air) of the volume, which made the visualization of the actual structures inside the datasets possible. The right images show the interface for the second level of interaction (with the color picker omitted) displaying the chosen thumbnails interactively. The opacity TF is also shown along with its respective resulting image, illustrating the difference between them as well as the maximum opacity given to a voxel in the volume.

Compared to other methods, specifically to Kindlmann and Durkin [9], Kniss et al. [11, 12] and Marks et al. [18], our approach differs from them by using data values to drive the definition of initial opacity TFs, and the possibility of assigning color during the second level of interaction, without the need of specifying (manually) a direct or indirect mapping in a graphic plot. Compared to Design Galleries [18], our method generates fewer 3D thumbnails, while that method produces several 2D ones. In relation to the work of Kindlmann and Durkin [9], our method employs a color TF while their work deals only with opacity TFs. In summary, our main objective was to ease the process of specifying transfer functions, allowing the user to quickly find an acceptable TF yet giving a high level control for the refinement of these functions.

5. Conclusions

An effective visualization must be capable of showing important regions from the volume without occluding them with uninteresting materials. This is generally a hard task when done through trial and error. In this paper we presented an interactive method and tool aimed at assisting the task of specifying opacity and color transfer functions for general purpose datasets. We not only combined but extended existing techniques to develop a more intuitive tool, which takes advantage of the possibility of automatic generating transfer functions, while giving suitable control to the user.

A possibility of future work is the specification of multidimensional TFs. Despite the improvement in classifying different materials which share the same range of data values, the task of specifying one-dimensional TFs is difficult, and becomes even worse with multidimensional TFs. Besides that, memory bandwidth is still a bottleneck in today's graphics hardware, and these functions demand a large amount of memory.

Another aspect for further research is the detection of features of interest. We can try other image processing filters to extract different materials from the volume, or even find the boundaries in a more accurate fashion. However, the first and second derivatives are very straightforward to implement and do not demand high computational cost, unlike other edge detection methods.

6. References

[1] Bajaj, C., V. Pascucci, and D. Schikore. The contour spectrum. In *Visualization '97*, pages 167-173, Phoenix, AZ, October 1997.

[2] Cabral, B., N. Cam, and J. Foran. Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware, In *Symposium on Volume Visualization*, 1994.

[3] Dietrich, C., L.P. Nedel et al. Real-time interactive visualization and manipulation of volumetric data using GPU-based methods. In *SPIE Medical Imaging – Visualization, Image-Guided Procedures, and Display*, vol. 5, n. 21, pages 181-192, San Diego, 2004.

[4] Engel, K., M. Krauss, and T. Ertl. High-quality Pre-integrated Volume rendering Using Hardware-accelerated Pixel Shading. In *Eurographics/SIGGRAPH Graphics Hardware Workshop*, pages 9-16, 2001.

[5] Hladuvka, J., A. Konig, and E. Groller. Curvature-Based Transfer Functions for Direct Volume Rendering. Technical Report, Vienna University of Technology, 2000.

[6] Hönigmann, D., J. Ruisz, and C. Haider. Adaptive Design of a Global Opacity Transfer Function for Direct Volume Rendering of Ultrasound Data. In *Visualization 2003*, pages 489-496, 2003.

[7] Kaufman, C., R. Bakalash, D. Cohen, and R. Yagel. A survey of architectures for volume rendering. *IEEE Engineering in Medicine and Biology Magazine*, 9(4): 18–23, December 1990.

[8] Kindlmann, G. Transfer Functions in Direct Volume Rendering: Design, Interface, Interaction. SIGGRAPH 2002 Course Notes, Course 50, 2002

[9] Kindlmann, G. and J. W. Durkin. Semi-automatic generation of transfer functions for direct volume rendering. In *Symposium on Volume Visualization*, pages 79-86, 1998.

[10] Kindlmann, G., R. Whitaker, T. Tasdizen, and T. Möller. Curvature-Based Transfer Functions for Direct Volume Rendering: Methods and Applications. In *Visualization 2003*, pages 512-520, 2003.

[11] J. Kniss, G. Kindlmann, and C. Hansen. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In *Visualization 2001*, pages 255-262, San Diego, CA, USA, October 2001.

[12] Kniss, J., S. Premoze, M. Ikits, A. Lefohn, C. Hansen, and E. Praun. Gaussian Transfer Functions for Multi-field Volume Visualization, In *Visualization 2003*, pages 497-504, 2003.

[13] König, A. and E. Gröller. Mastering Transfer Function Specification by Using VolumePro Technology. In *Spring Conference on Computer Graphics 2001*, volume 17, pages 279-286, April 2001.

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

[15] Levoy, M. Efficient Ray-Tracing of Volume Data. *ACM Transactions on Graphics*, 9(3): 245–261, 1989

[16]Lichtenbelt, B., R. Crane, and S. Naqvi. *Introduction to Volume Rendering*, Prentice Hall, Upper Saddle River, NJ, 1998.

[17] Lorensen, W. and H. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *Computer Graphics*, 21(4):163-169, 1987.

[18] Marks, J. et al. Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation. In *SIGGRAPH' 97*, pages 389-400, Los Angeles, CA, August 1997.

[19] Meissner, M. U. Hoffmann and W. Strasser. Enabling classification and shading for 3D texture mapping based volume rendering using openGL and extensions. In *IEEE Visualization'99*, pages 207-214, San Francisco, 1999.

[20] Meissner, M., H. Pfister, R. Westermann, and C. Wittenbrink. Volume Visualization and Volume Rendering Techniques. Eurographics 2000 Tutorials #6, Interlaken, Switzerland, 2000.

[21] Pekar, V., R. Wiemker, and D. Hempel. Fast Detection of Meaningful Isosurfaces for Volume Data Visualization. In *IEEE Visualization 2001*, pages 223-230, San Diego, CA, USA, 2001.

[22] Pfister, H., J. Hardenbergh, J. Knittel, H. Lauer, and L. Seiler. The VolumePro Real-Time Ray-Casting System. In *SIGGRAPH'99*, pages 251-260, August 1999.

[23] Pfister H. et al. The Transfer Function Bake–off. *IEEE Computer Graphics and Applications*, 21(3): 16-22, May/June 2001.

[24] Schroeder, W. et al., *The Visualization Toolkit,* Kitware, Inc. Prentice Hall, 1998.

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

[26] van Gelder, A. and K. Kim. Direct volume rendering with shading via 3D textures, In *Symposium on Volume Visualization*, pages 23-30, San Francisco, CA, October 1996.

[27] Yagel, R. Volume Viewing Algorithms: Survey. In *International Spring School on Visualization*, Bonn, 2000.

[28] http://www.inf.ufrgs.br/~jlprauchner/sibgrapi2005/

7. Acknowledgments

We are grateful to Steve Callahan, from the University of Utah, for his careful revision of the first version of this paper. We also acknowledge the support from the Brazilian funding agencies CNPq and FAPERGS.

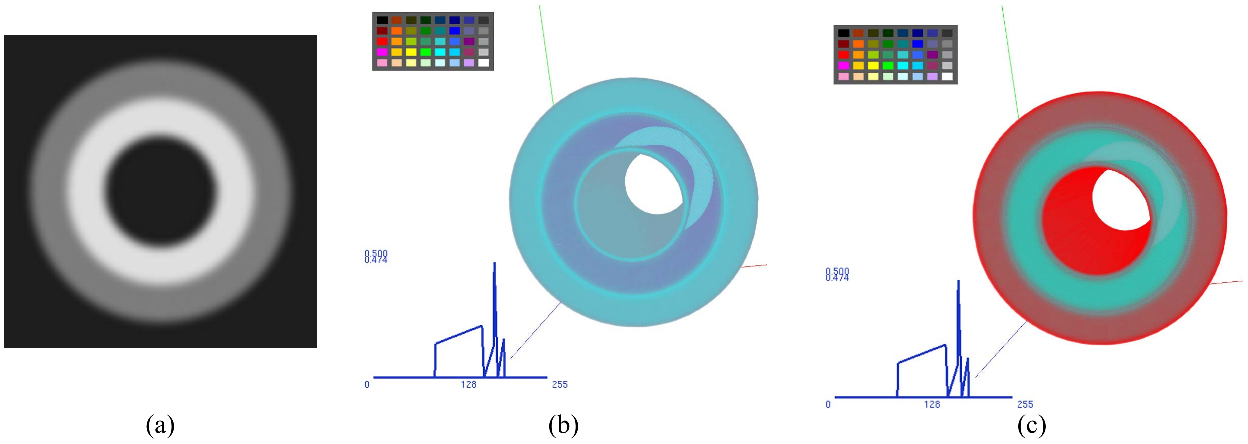

Figure 4. Rendering of the synthetic cylinder. (a) Sample slice; after choosing one of the thumbnails, the center image (b) is displayed; changes in the color of two materials (using the color picker) result in the rightmost image (c).

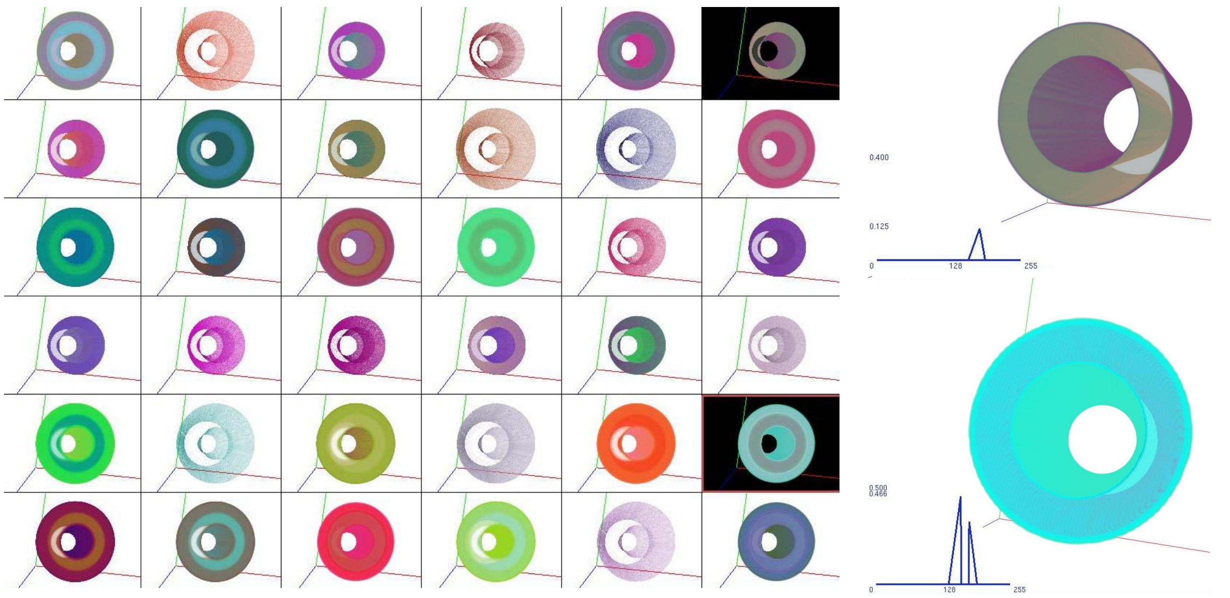

Figure 5. Rendering of the cylinder dataset. Left: thumbnails are rendered; two images were chosen and saved (shown in dark background). Right: interface for the second level of interaction, showing the chosen images together with their opacity TF. Dataset is 64^3.

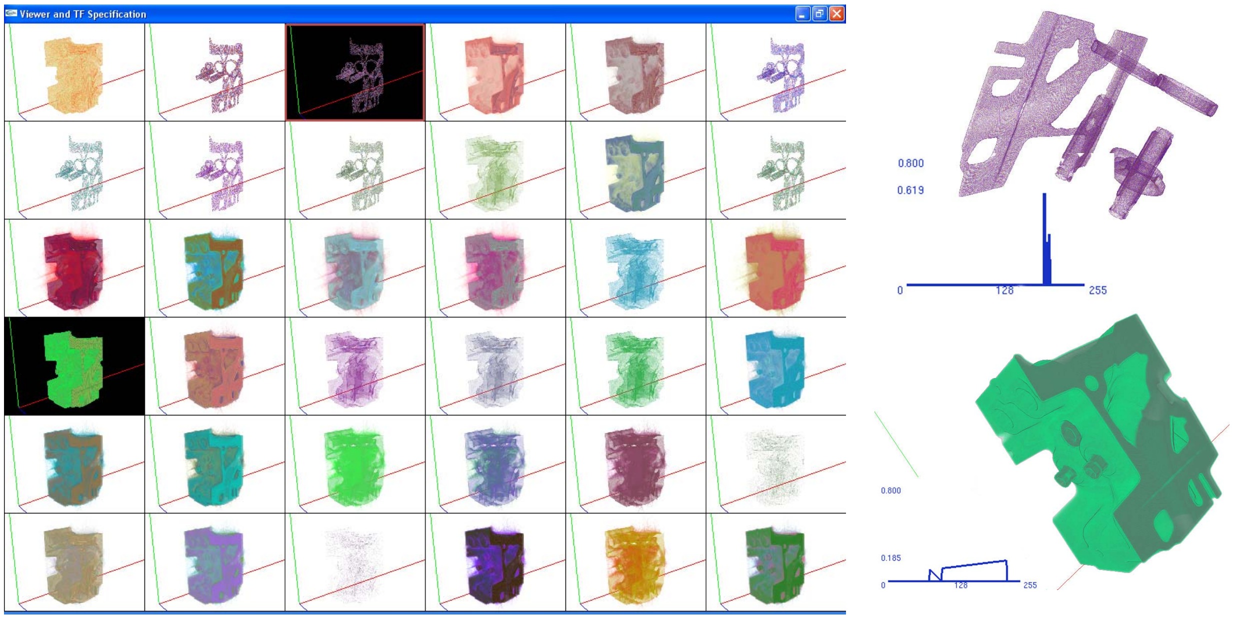

Figure 6. Rendering of the engine block dataset. Left: interface with the thumbnails; two images were chosen (shown in dark background) and saved. Right: interface for the second level of interaction, showing the chosen images together with their opacity TF. Dataset is 256^3.

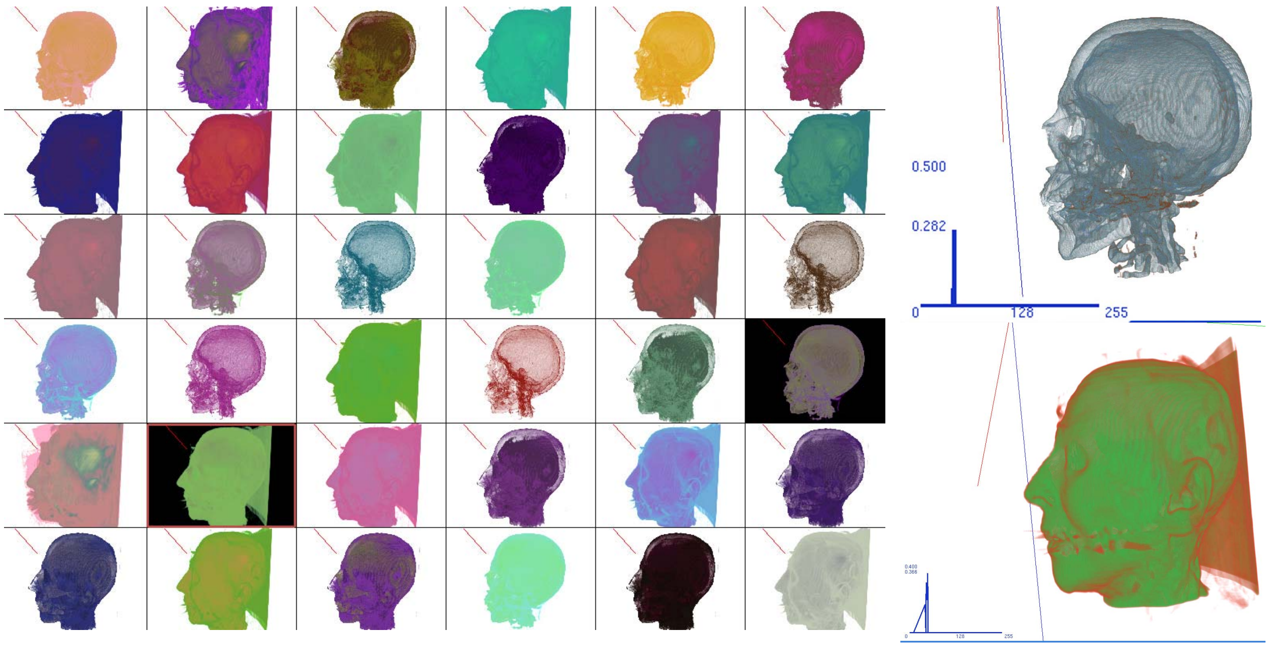

Figure 7. Rendering of the UNC CT Male. Left: interface with the thumbnails; two images were chosen (shown in dark background) and saved. Right: Interface for the second level of interaction, showing the chosen images along with their opacity TF. We can notice that the system generates images with different transparency levels and colors. Dataset is 128^3.