Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Volume visualization and exploration through flexible transfer function design [☆]

Francisco de M. Pinto [*], Carla M.D.S. Freitas

Instituto de Informática, UFRGS, Av. Bento Gonalves, 9500 - Campus do Vale - Bloco IV, Caixa Postal 15064, CEP 91501-970, Porto Alegre, RS, Brazil

A R T I C L E I N F O

Article history:
Received 31 October 2007
Received in revised form
10 April 2008
Accepted 13 April 2008

Keywords:
Transfer function
Direct volume rendering
Volume data
Volume visualization

A B S T R A C T

Direct volume rendering (DVR) is a well-known method for exploring volumetric data sets. Optical properties are assigned to the volume data and then a DVR algorithm produces visualizations by sampling volume elements and projecting them into the image plane. The mapping from voxel values to optical attribute values is known as transfer function (TF). Therefore, the quality of a visualization is highly dependent on the TF employed, but its specification is a non-trivial and unintuitive task. Without any help during the TF design process, the user goes through a frustrating and time-consuming trial-and-error cycle. This paper presents a useful combination of TF design techniques in an interactive workspace for volume visualization. Our strategy relies on semi-automatic TFs generation methods: boundary emphasis, stochastic evolutive search in TF space, and manual TF specification aided by dual domain interaction. A two-level user interface was also developed. In the first level, it provides multiple simultaneous interactive visualizations of the volume data using different TFs, while in the second one, a detailed visualization of a single TF and the respective rendered volume are displayed. Moreover, in the second level, the TF can be manually refined and the volume can be further inspected through geometric tools. The techniques combined in this work are complementary, allowing easy and fast TF design and data exploration.

1. Introduction

Volume rendering is widely known as a set of methods for visualization of large three-dimensional (3D) scalar or vector fields, mainly in medical and scientific data exploration. In these areas, one often deals with 3D images, such as those obtained from CT and MRI devices or numerical simulation data. Volume rendering techniques and algorithms are well described in the literature [1], and can be classified as isosurface extraction methods and direct volume rendering (DVR) methods. The former methods extract polygonal meshes representing isosurfaces in the volume and then use the traditional rendering pipeline to display the meshes (see [2], for a well-known example). On the other hand, DVR methods display volume data without extracting an intermediate geometry. DVR and its advantages were first described by Levoy [3]. Modern graphics hardware supports volume rendering at interactive rates using either of these approaches.

To obtain useful images through DVR, voxels have to be classified in order to determine which ones must be displayed. This classification is typically performed by transfer functions (TFs), which associate values of optical attributes to voxels based on their values. Opacity and color are the most common optical properties used in TFs. The degree of opacity can make a voxel more or less visible and is normally used to emphasize voxels in boundaries between different homogeneous regions of the volume [4]. Other optical properties may also be used, such as specular reflection coefficients [5], spectral reflectance [6] and light scattering coefficients [7]. More recently, the concept of style TFs was introduced by Bruckner and Gröller [8], who used TFs to define the rendering style of volume regions based on data values and eye-space normals. The information conveyed by an image built from volume data is, therefore, highly dependent on the quality of the TF. However, TF design is a non-trivial and unintuitive task, and has been referred as one of the top 10 problems in volume visualization [9].

DOI of original article: 10.1016/j.cag.2008.04.004

[☆] This paper was mistakenly published in volume 32(4). It is reprinted here in the Computer Graphics in Brazil issue it belongs to.

[*] Corresponding author. Tel.: +55 51 3308 6811.

E-mail addresses: fmpinto@inf.ufrgs.br (F.M. Pinto), carla@inf.ufrgs.br (C.M.D.S. Freitas).

One-dimensional (1D) TFs take into account only scalar voxel values, and are the most common TFs, although having a limited classification power. On the other hand, multi-dimensional TFs allow more freedom in voxel classification by taking as arguments vectorial values or combinations of local measures of scalar fields, such as derivative values [10,11], neighborhood, position [12], curvature [13,14] and statistical signatures [15]. Notwithstanding, the design complexity grows with the size of the TF domain [10], and the memory required to implement truly multi-dimensional TFs restricts their application [16]. In this work, we adopted 1D TFs due to their simplicity and low-memory requirements, since they can be implemented as small lookup tables. Furthermore, the pre-integrated volume rendering technique, proposed by Engel et al. [17], allows high-quality DVR at interactive rates using 1D TFs.

Designing TFs with no assistance leads to trial-and-error efforts. Therefore, several automatic and semi-automatic techniques for specifying TFs have been proposed [4,8,15,18–23]. They can be guided by analysis of the volumetric data (data-driven) or analysis of the generated images (image-driven) [9]. In any case, to make the process less frustrating and less time-consuming, the user must be given a rapid feedback with real-time rendering frame rates.

The main contribution of our work is a two-level interface method that combines a set of useful tools for semi-automatic 1D TF design and fast data exploration in an interactive workspace. The first level of our interface presents several thumbnails of the volume data rendered with different TFs, allowing immediate insight of the main structures in the data set. The second level shows a detailed visualization of a single TF as well as the resulting rendering. TFs can be easily generated and refined using semi-automatic boundary emphasis, stochastic evolutive search and manual design aided by dual domain interaction [11]. These three approaches are complementary and were successfully combined in this work, improving the idea of two-level interaction proposed by Prauchner et al. [24].

This paper extends our previous work [25] with an improved interface and an experimental evaluation of our approach. We implemented a history tree that keeps track of the TF evolution and allows the user to go back to a previously specified TF. The evaluation was performed as an experiment with 15 subjects who accomplished two visualization tasks with different data sets. This way we tested the usability of our methods with potential users. We also implemented a set of geometric tools for volume inspection inspired by the work of Dietrich et al. [26], but their description is beyond the scope of this article.

The paper is organized as follows. Section 2 discusses the closest related works. Section 3 describes the proposed interface and the TF design techniques provided within it. Implementation details are addressed in Section 4, while in Section 5 we present the evaluation of our tools. At last, in Section 6, we draw some conclusions and point directions for future work.

2. Related work

The TF specification problem has received much attention from researchers. Traditional approaches rely on user effort in adjusting control points of a graphic plot of the TF [27]. The control points—scalar values associated with values of optical attributes—are then interpolated in order to build the TF. However, with no clues or prior knowledge about the data, this is a “blind process”. Some data-driven approaches provide users with higher-level information [18,23] that helps in obtaining insight about the data distribution, thus supporting manual TF design. Some methods hide the TF from the user through abstractions: Tzeng et al. [28] implemented multi-dimensional TFs as neural networks or support vector machines, providing users with a simple interface for selection of voxels as training sets for the learning process; He et al. [21] used genetic algorithms to perform stochastic evolutive TF design, requiring the user to only choose the best rendered images, i.e., the best TFs. Other methods offer a simplified space for TF specification. Rezk-Salama et al. [29] created models of TFs that are carefully adjusted by specialists for several data sets of the same type in order to reveal the desired structures. Then, they applied PCA to represent the parameter set of each model by a single variable with an associated semantic. The models can be reused for new data sets by setting only that variable. Šereda et al. [30] used hierarchical clustering to group voxels according to their LH signatures [31]. The user navigates through the hierarchy searching for the branches corresponding to regions of interest. LH signatures also constitute a two-dimensional (2D) space for definition of TFs where boundaries in the data set are easily classified. Bruckner and Gröller [8] represent pre-defined shading styles as lit spheres that the user can choose to build rendering style TFs. This technique allows the user to obtain TF-guided non-photorealistic visualizations of volumes. We have also worked on multi-dimensional TFs specification [32], using self-organizing maps to perform non-linear dimensional reduction of multi-dimensional voxel values, creating an abstraction of the nD TF domain in an easy-to-use interface.

Kindlmann and Durkin [4] proposed a derived space for specification of opacity TFs in which the user assigns opacity levels to voxels as a function of the distance between the voxel and the nearest boundary. The authors assume that boundaries are smoothed by a Gaussian filtering process due to the Gaussian frequency response of 3D scanners. Informative histograms are built relating voxel scalar values to first and second derivative values. From these histograms, the mean first and second derivative values associated with each voxel value are used to estimate the distance to the nearest border. Boundaries often need to be emphasized, thus voxel values with small estimated distances are associated with large opacity values.

The design galleries method [22] is based on the generation of a wide set of TFs through a dispersion algorithm, in a pre-processing phase. Then, the obtained TFs are used to render thumbnails that are grouped by similarity and presented in an interface where the user can pick and zoom in the most appealing thumbnails to observe the resulting image. The stochastic approach for TF design proposed by He et al. [21] also uses galleries of thumbnails. The authors represent TFs as vectors of control points and the smooth interpolation of the control points produces the actual TF. The method employs genetic algorithm operators to create new TFs from the previous ones. The mutation operator changes control points, while the crossover operator produces new TFs by concatenating sub-vectors of control points from different TFs. Given a set of TFs, and their respective rendered images presented as thumbnails, the “best” ones are selected as parents, either by the user or through automatic evaluation based on objective image processing metrics. The initial TFs are randomly specified and the TF population evolves as the parents are selected, and new TFs are generated by applying the operators mentioned above to the current parents.

Prauchner et al. [24] used Kindlmann and Durkin’s method [4] to classify the voxel values by the estimated distance to the nearest border. The voxel scalar values with the smallest distances are selected and random subsets of these values are then built. The values in the subsets are used as control points which receive random color and random opacity value. Each TF is obtained by interpolating the control points in one of those subsets. The generated TFs are used to render a gallery of thumbnails of the volume data, similarly to the design galleries method. This is the first level of the two-level interaction interface proposed by

Prauchner et al. [24]. In the second level, the user can visualize a selected thumbnail in high resolution and refine its TF by manually adjusting the control points. The thumbnails can be re-generated at any time using new TFs.

Our method [25] improves the TF specification process by combining features from the approaches referred above into a general-purpose interactive workspace that implements image-driven and data-driven TF design through a two-level interface. The set of features provided by our interface is described in the next section.

3. TF specification

Most researchers agree that TF specification methods should not overload the users nor exclude them from the process [9]. The quality of a TF depends on the amount of information conveyed by the generated image—a subjective metric. Therefore, it is hard to automatically evaluate how “good” a TF is. Fully automatic TF specification methods may miss important features of the volume while completely manual TF design may demand a lot of effort and time, mainly from users that do not have prior knowledge about the data. In this section we outline some features that are desired in TF design tools and then we present our method.

3.1. Requirements for TF design tools

A TF design interface should offer useful information about the data as well as guidelines for the TF building process. The interface may also suggest sets of TFs based on heuristics or mathematical criteria, but the choice between the alternatives should be made by the user. Abstractions of the TF specification process can also be applied with success as shown by Tzeng et al. [12] and Rezk-Salama [29]. In any case, the user makes decisions and sees the results; hence immediate feedback is needed to make the process continuous, with minimum cognitive effort.

Experienced users working with known data can find good TFs with relatively few interactions with a simple manual TF editing interface. However, when the data are unknown, a trial-and-error approach may not be a good choice. In these cases, a two-level interface for TF specification like the one proposed by Prauchner et al. [24] is very useful. The different thumbnails presented in the first level allow an immediate insight of several volume structures. However, the potential benefits of a two-level interface were not thoroughly explored in that work. The TFs of the most appealing thumbnails selected from the set presented in the first level of interaction may also be used as a basis to generate other TFs. Moreover, the manual refinement should be more flexible and the interface must allow interaction in both TF and spatial domains in order to improve the understanding of their correlation. Changes in the TF must have immediate effect in the rendered volume, and voxel values queried in the spatial domain must be also shown in the TF domain [11]. We try to fulfill all these requirements with our two-level interface.

3.2. Two-level interaction overview

Our method combines three techniques for TF design: boundary emphasis, stochastic search and manual specification with dual domain interaction. The two-level interaction is summarized in Fig. 1.

User interaction starts at the first level, where thumbnails generated initially through the boundary emphasis technique are

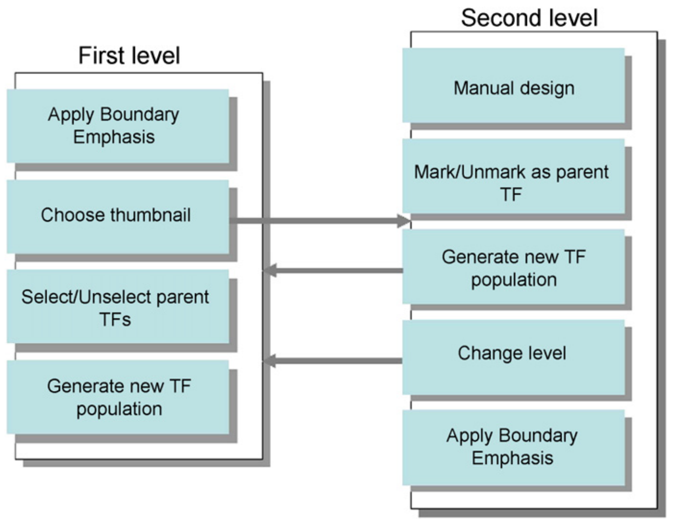

Fig. 1. The main actions allowed in each level of the interface, the arrows representing level changes.

shown. At this level, the user can generate new TFs (new thumbnails) by either reapplying boundary emphasis or selecting thumbnails as parents and then applying stochastic search. The user can also select a single thumbnail, action which leads to the second level of interaction, where he/she can apply boundary emphasis and manual design only to the selected TF. Any changes in the TF selected for the second level do not affect the other TFs exhibited in the first level. The user can change back to the first level anytime. The TF design techniques combined in the proposed interface are described in the next subsections.

3.3. Boundary emphasis

The initial set of TFs for rendering the thumbnails in the first level of interaction is generated using a boundary emphasis technique, based on a 3D histogram that relates voxel values to the first and second derivatives, following Kindlmann and Durkin [4]. Like Prauchner et al. [24], we use the histogram approach to choose control points for the TFs. The control points correspond to voxel values that are probably in a boundary region, and receive opacity values chosen randomly between a maximum and a minimum value. When building the TF lookup table, an interval of voxel values around each control point is assigned with non-zero opacities in a decreasing manner, depending on the distance from the control point, which builds triangular shapes in the opacity TF. The width of the intervals is adjustable. The color TFs are also designed to emphasize boundaries. Each control point is associated with two different random colors: one for voxel values larger than the control point’s value, and another for the smaller ones. Between two control points, the color values are linearly interpolated. This way, boundary regions present color discontinuities, as suggested by Fujishiro et al. [33]. Fig. 2 shows the benefits of this method for color TF generation as well as the triangular shapes in the opacity TF centered at boundary values. In the TF graphic plots, voxel values are represented as the horizontal axes, while opacity values are represented as the vertical axes. The color TF is directly represented.

3.4. Stochastic evolutive TF design

In order to generate the TFs employed for rendering thumbnails for the first level of interaction, we not only use the

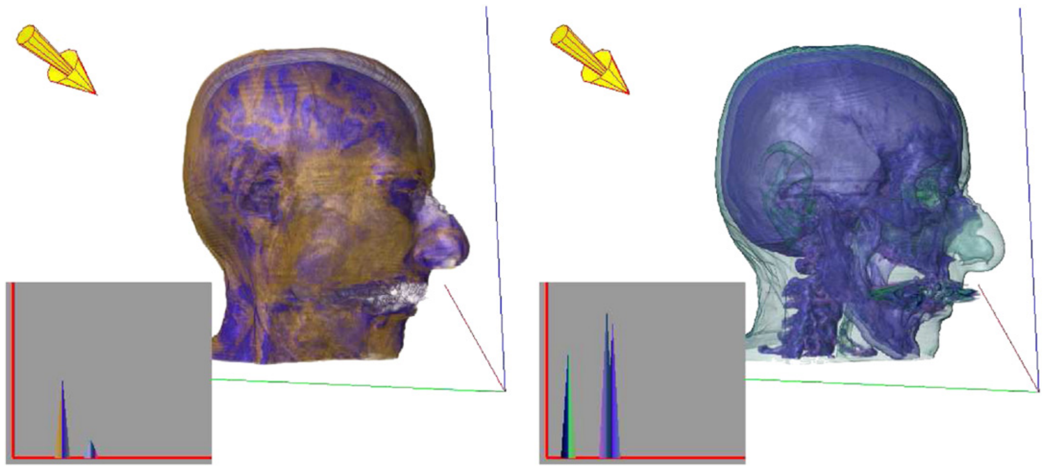

Fig. 2. The cadaver data set rendered using TFs generated by our boundary emphasis algorithm. The employed TF is displayed in the lower left corner of each image. The triangular shapes in the opacity TF and the color discontinuities at specific voxel values are responsible for emphasizing boundaries. In the left image, the color discontinuity allows clear visualization of the saliencies in the outermost part of the brain despite the similarity between the voxel values. The arrows indicate the incident light direction.

histogram approach by Kindlmann and Durkin [4], but also a more flexible implementation of the technique proposed by He et al. [21]. While their implementation of genetic algorithms codifies TFs as vectors of control points taken as genes, our codification of a TF is the TF lookup table itself, where the entries are the genes. This way we can easily combine the stochastic search with any other method for 1D TF design.

The method works as follows. Given a set of thumbnails and associated TFs, the user selects the best ones to be the parents of a new population, as shown in Fig. 3. Then, the TFs produce descendants by mutation, crossover or both, in a random way, and a new set of TFs (and thumbnails) are obtained, preserving only the parents. By selecting and deselecting thumbnails of the new set (population), the user can define the parents of the next TF population. If the quality of a particular thumbnail could not be properly evaluated due to its low resolution, the image can be better inspected in the second level of interaction, which shows a high resolution rendering of the volume. The crossover and mutation operators are described below.

- *Crossover*—crossover operator randomly takes two TFs among those selected as parents and cuts them at two points, also randomly determined. The interval between the two points of one TF is combined with the two outer intervals (defined by the same two points) of the other TF, as shown in Fig. 4a. In the graphic plot, the horizontal axis represents voxel values and the vertical one represents opacity. The vertical black lines are the two cutting points.
- *Mutation*—mutation operator can be applied repeatedly to a TF. Fig. 4b shows a TF and three types of mutation (M1, M2 and M3) applied to regions defined by pairs of points marked as vertical black lines. M1 is a triangular mutation: an opacity value is defined for the center of the mutation and a linear interpolation is applied between the central opacity value and the opacity values at each border of the mutation. M2 is a random mutation: the opacities are randomly set for the voxel values around the mutation center. M3 is a rectangular mutation. The center and the range of the mutations as well as colors and opacities are randomly set between limited intervals. There is also a fourth type of mutation: a random global scale applied to all opacity values.

The genetic operators produce high frequency features in the TF domain, which might cause artifacts in the resulting images. Since we implemented Engel's pre-integrated volume rendering method [17], which deals very well with high frequencies, those features do not affect the images rendered by our tool.

3.5. Manual specification

We propose a simple interface for manual specification of TFs at the second level of interaction. The user draws directly on the TF graphic plot. By dragging the mouse with the left button pressed, the user draws a line, setting the shape of the opacity TF, as shown in Figs. 5(a)–(c). Using the mouse right button, instead, the user sets an interval of influence in the voxel value axis. The selected interval is affected by color selection or scale changes in the opacity values. Once an interval is set, the user can select a color from a color picker. The interval is painted with this color linearly interpolated with the existing ones, as shown in Figs. 5(d) and (e). The center of the interval receives the selected color and the contribution of that color decreases toward the borders of the interval. Using the keyboard, the user can scale the opacities in the selected interval (see Fig. 5f). During manual TF modification, the voxel value under the cursor is shown in a small box (Figs. 5(b) and (c)).

When users are able to make correlations between TF domain and spatial domain, manual TF design becomes simpler. Presenting colors in the TF graphic plot helps in this way. The colors used to render the volume are also used to paint the TF plot, thus providing information about the distribution of voxel values in the volume (spatial domain). In addition, we implemented a volume investigation scheme that can also be used to edit the TF, as a dual domain interaction approach. The user can use a clipping plane to sweep the volume, as shown in Fig. 6. The slice sampled by the clipping plane is displayed at the top right as a gray level image directly showing the voxel values. As the mouse pointer moves on the clipping plane or on the slice, the value of the pointed voxel is presented, and a small white square is displayed on the TF graphic plot, indicating the position of that voxel value in the TF domain. By clicking using the mouse left button, the opacity value associated with that voxel value is increased. This scheme allows direct volume inspection and is very useful to emphasize isosurfaces passing through a specific point of the volume.

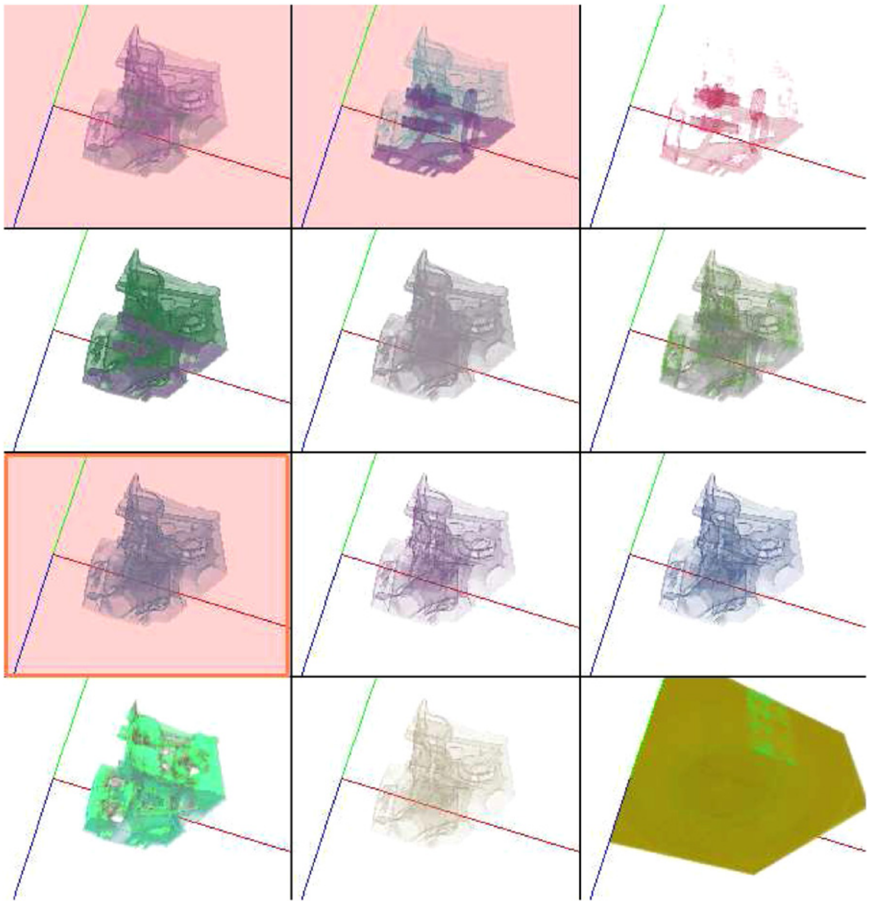

Fig. 3. Selection of parents for a new population of TFs in the first level of interaction. Selected thumbnails are presented in red background. The volume is the well-known engine data set.

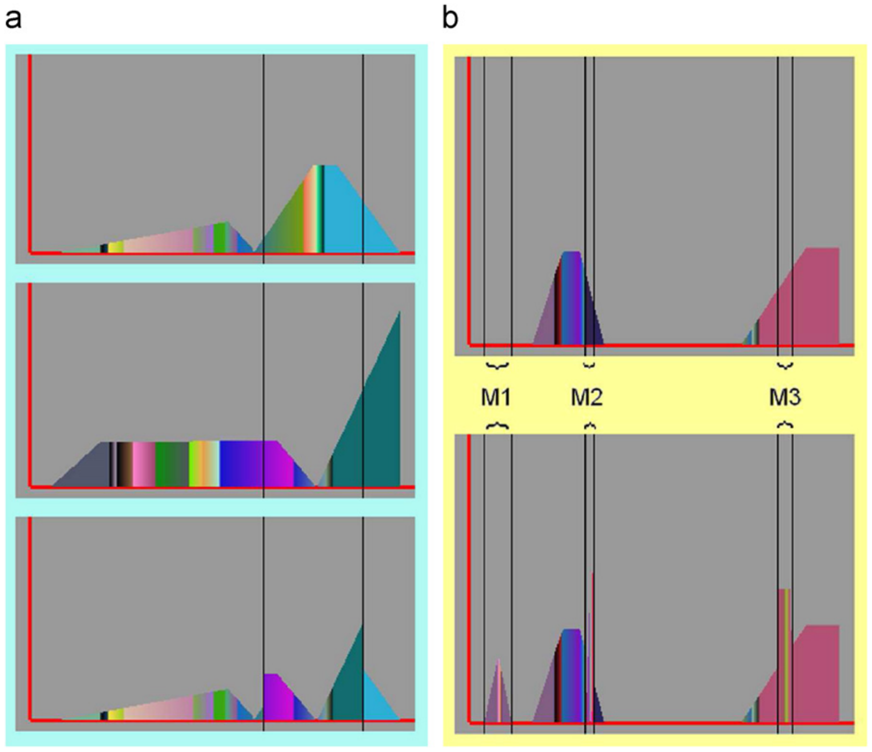

Fig. 4. (a) The crossover operator. The topmost two TFs are the parents and the bottom one is the resulting TF. (b) Three types of mutations applied to a TF. The bottom TF is the result.

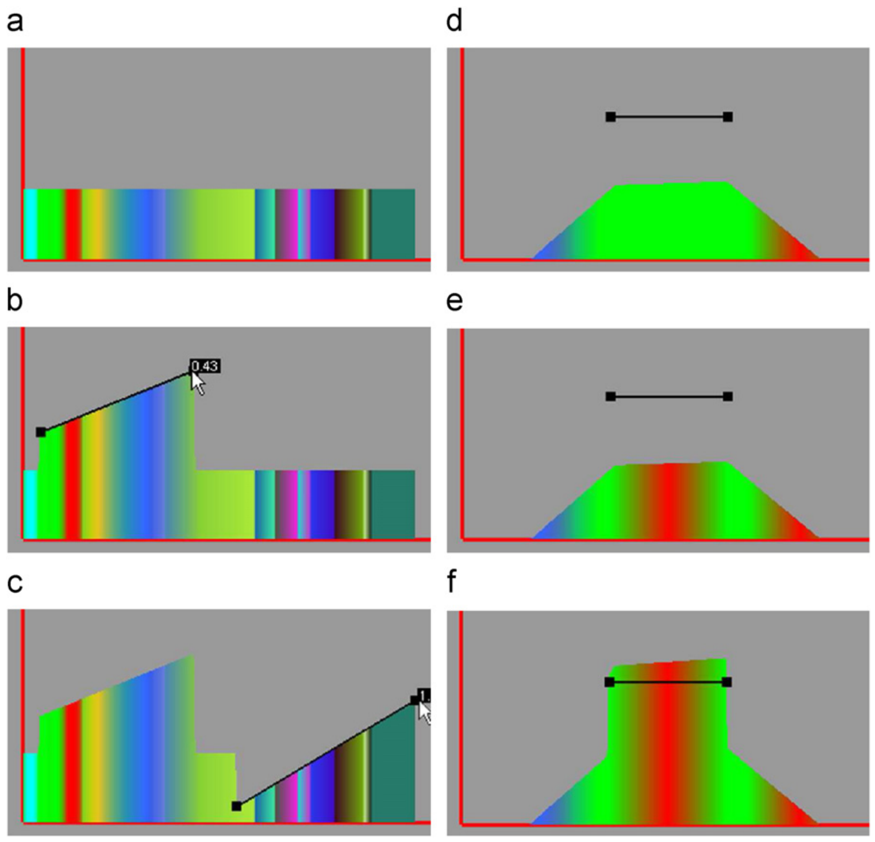

Fig. 5. A TF (a) and the results of two design steps (b and c). A second TF (d) with a selected interval of influence; red is assigned to the interval (e) and the opacities for the same interval are scaled (f).

3.6. *History tree*

As discussed by Bavoil et al. [34], it is important to keep track of the evolution of a visualization. In their work, changes made to the pipeline and parameters during the process of building an expressive visualization are recorded to allow retrieving the previous versions as well as to reproduce results later on. This way they also provide a manner to reuse configurations for visualizing different data sets.

In our work we have implemented a persistent history tree of the TFs used to produce visualizations. In the second level of interaction the user is able to save the current TF. The TF is then stored as a node of the tree, together with a snapshot of the respective rendered volume. The history tree is displayed in a separate window and each node is graphically represented by the snapshot taken from the visualization. Any saved TF can be retrieved by clicking on the respective tree node. The tree (see Fig. 7) can be scaled and translated by dragging the mouse and its layout is automatically maintained. The user can also delete branches of the tree when it becomes too large.

4. Implementation details

Our volume rendering tool was implemented in C++ using GLUT [35] and GLUI [36] libraries for the interface (see Fig. 8), and OpenGL [37] and CG [38] for volume visualization. The rendering algorithm runs in GPU and is based on 3D texture sampling using view-aligned slices as proxy geometry. The number of slices can be changed by the user, and is automatically reduced when the volume is being rotated to guarantee interactive rates in both levels of interaction. It is worthy to mention that Engel's pre-integrated volume rendering method [17] allows high-quality rendering even with a relatively small number of proxy planes.

In our approach, the volume data are stored in the GPU memory as a 3D texture with eight bits of precision. TFs are represented as lookup tables with 256 entries and pre-integrated tables are stored as 2D RGBA textures. We can pre-integrate color and opacity contributions neglecting color attenuation inside volume slabs, which is much faster than full pre-integration. We apply this simplified pre-integration in the first level, since an action can modify several TFs (36 TFs with respective thumbnails in our current interface). In the second level of the interface a single TF is under modification, and then we apply full pre-integration without compromising the interactivity. Volume shading can be enabled or disabled by the user and is based on per-fragment Phong lighting using the gradient of the scalar field as normal vectors.

5. Evaluation and discussion

In this section we describe the experiment performed in order to evaluate our TF design technique and then we present the results. Assuming that our manual TF design tool with dual domain interaction would be at least as good as the traditional interfaces based on adjustment of control points, we wanted to prove the usefulness of our interface by comparing the time and interaction steps needed to build suitable visualizations of selected data sets using all resources of the workspace with the corresponding measurements taken when using only the manual design tool.

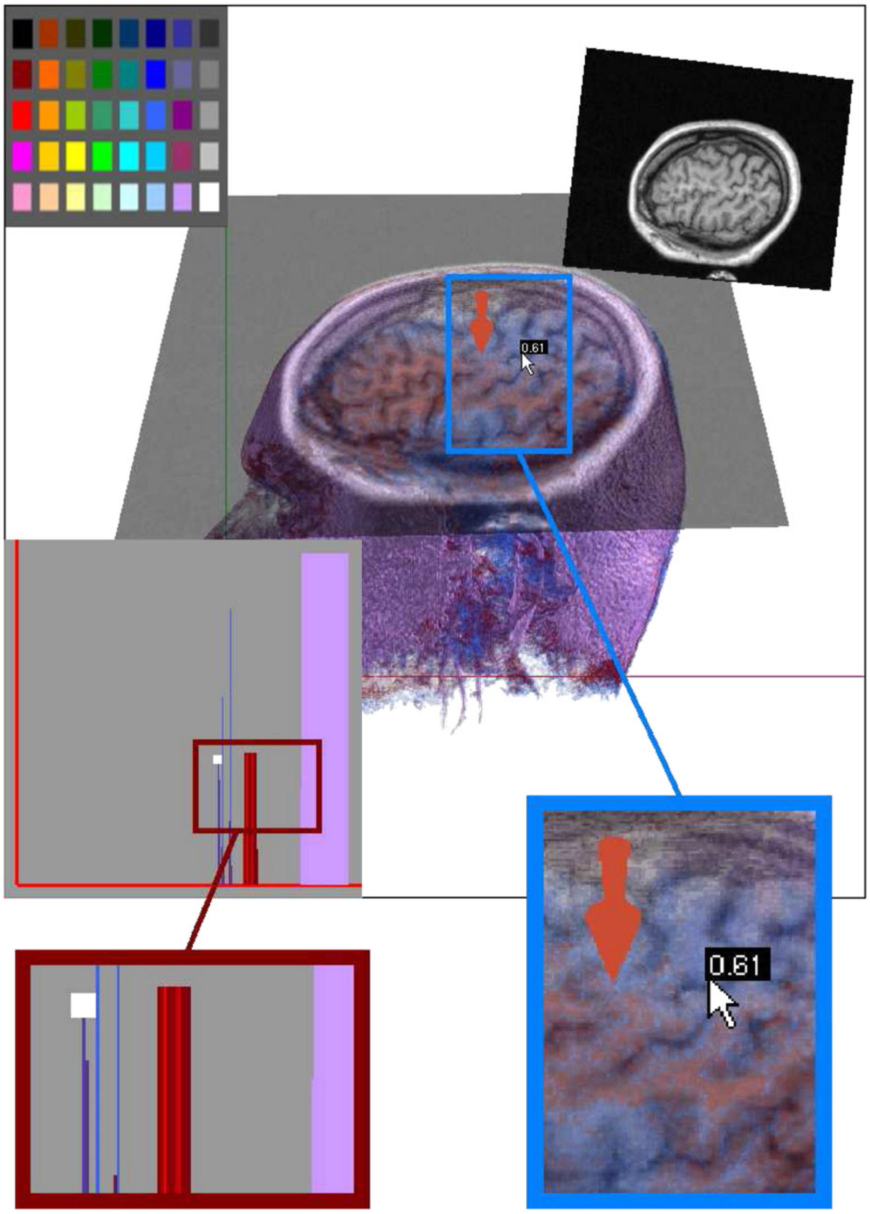

Fig. 6. A clipping plane is sweeping the volume, the red arrow representing its normal. The voxels on the clipping plane are displayed as a grayscale slice at the top right. The mouse pointer is querying a voxel on the plane (blue rectangle). The white square in the TF graphic plot (see the enlarged red rectangle) represents, in TF domain, the value of the queried position.

5.1. Subjects and procedure

Our subjects were 15 students, 10 of them having no experience in volume visualization and TF specification. Among the five experienced students, two had already developed their own volume visualization tools as well as TF design schemes; the other three were familiarized with volume visualization tools.

The experiment consisted of a training step, in which the subjects used our framework to visualize the Visible Male Head data set (Figs. 9 (a) and (b)), followed by two visualization tasks, one using all resources of the interface (task named as Full) and one using only the manual design tool and the clipping plane (task named as Manual). All tasks consisted in obtaining visualizations of two data sets (Chest and Cadaver Head), revealing the volume structures shown in Figs. 9 (c)–(f). For each task we presented one of these four images and asked the users to produce a visualization of the depicted structure by designing a proper TF. The subjects had never visualized those two data sets and they were instructed to stop the TF design process when they obtained a good visualization, according to their judgment. It was randomly decided whether a subject would start with the Full visualization task, using all framework's resources, or the Manual task, using only manual TF design and clipping plane. However, to prevent the order effect, we guaranteed that half of the users accomplished Manual first and the other half started with Full. The tasks assigned to the subjects were also randomly chosen among the visualizations of the four volume structures mentioned above.

During the experiments we recorded all interaction steps performed by the subjects, including the time spent in each step and the switching between the two levels of our interface. Two subjects were not able to accomplish the visualization task employing only manual TF specification and stopped after several minutes. After accomplishing the tasks, all subjects answered some questions about the TF design tools provided by our framework, as a subjective evaluation of their usefulness, and gave suggestions for improvements.

5.2. Results and discussion

Most subjects agreed that the interface's first level provides immediate understanding of the most important structures in the data sets. Also according to the subjects' comments, the boundary emphasis tool produces, at least, a good starting point, and the stochastic search is very useful for combining TFs that emphasize different structures, exploring new possibilities and refining TFs. They also agreed that our framework is simple to use and fast, due to real-time rendering and interaction.

We also asked the subjects which framework's tools or features they found most useful. Most of them (12 subjects) mentioned combinations of features, being solely manual design cited by only one. Multiple visualizations (first level) were mentioned by 60% of the subjects as one of the most useful features while stochastic search was cited among the most useful techniques by 73% of them. The clipping plane tool and the boundary emphasis feature were mentioned by 47% and 33% of the subjects, respectively. Only 13% of the subjects reported pure manual design among the most useful tools.

Table 1 summarizes the results obtained from the analysis of recorded data. For accomplishing Manual task, subjects spent 09 min 10 s in average, with a high standard deviation (05 min 22 s), while the Full task lasted 08 min 01 s in average, also with a high standard deviation (05 min 05 s). Although for more than half of the subjects the time spent during Manual was higher than during Full, *t*-test showed that there is no significant difference between these two sets of time measurements. However, *t*-test performed with data on number of interaction steps showed significant difference: users performed 185.5 steps in average during Manual task (standard deviation $= 168.2$), and only 96.5 steps (standard deviation $= 61.6$ steps) during Full task ($p = 0.011, \alpha = 0.05$). For most of the subjects, the number of interaction steps was reduced in 60% in average.

The detailed recording of interaction steps allowed us to verify that manual TF refinement is still a bottleneck, and that was the cause of the lack of significant difference in time results. Data showed that even when using all framework's resources, most of the interaction performed by the subjects was due to the manual refinement phase (see Table A1, in the Appendix, for detailed

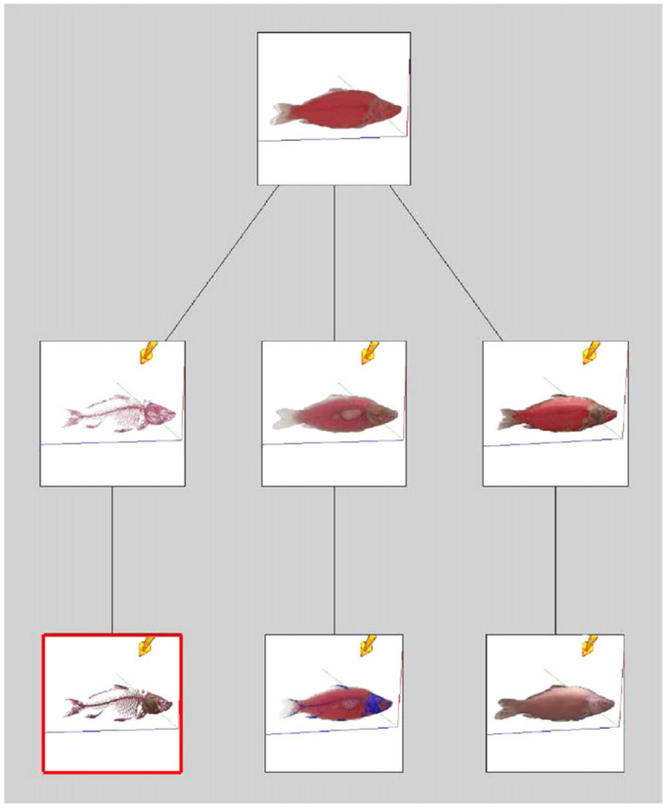

Fig. 7. The tree maintains the history of modifications along the TF design process. The red square indicates the current transfer function.

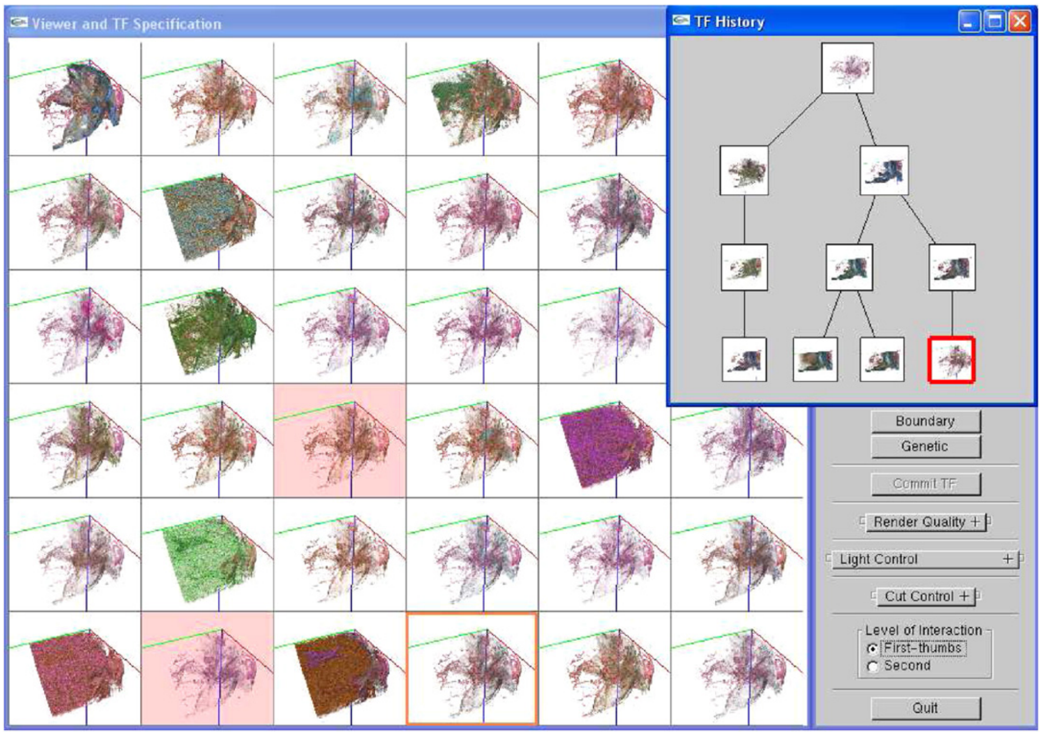

Fig. 8. A snapshot of the interface of our framework. Three windows are shown: the rendering window; the history window and the control window.

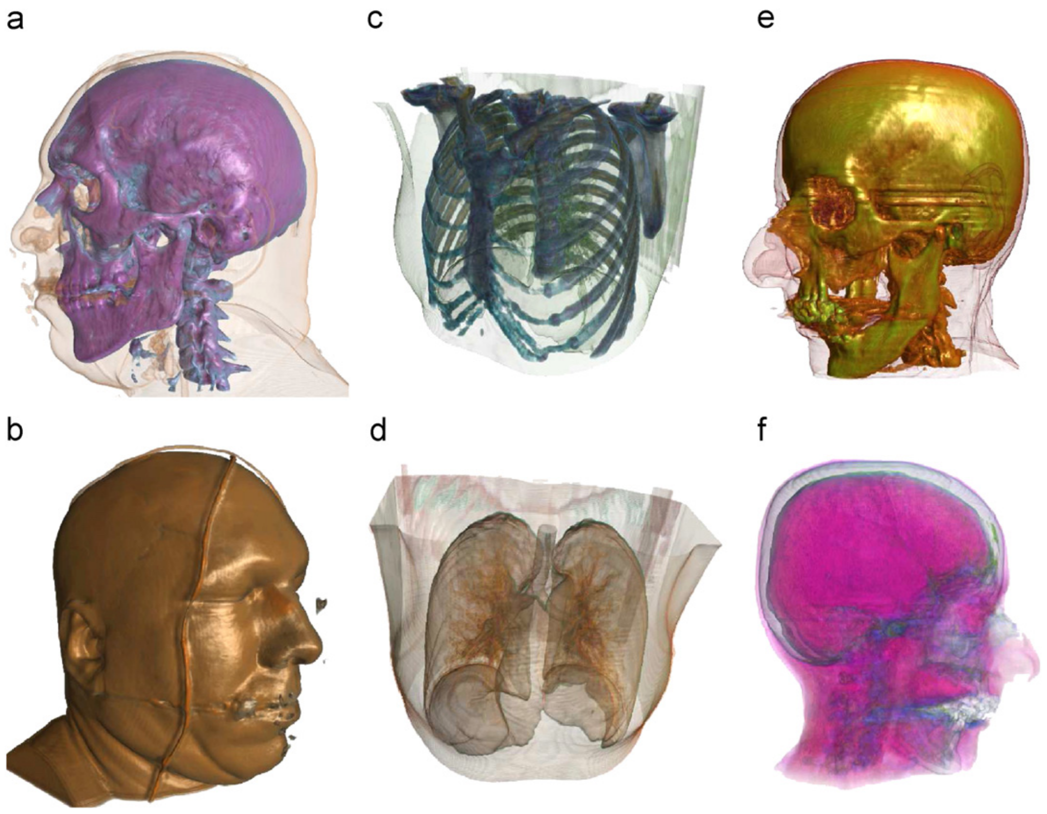

Fig. 9. Visualizations produced with our framework illustrating the three data sets used in the experiment: the Visible Male Head data set (a and b) used for training, the Chest data set (c and d) and the Cadaver Head data set (e and f) used in the proposed tasks.

Table 1
Average and standard deviation of time and number of interaction steps that the subjects spent to accomplish the two visualization tasks (Manual and Full).

	Manual		Full	
	Time (min:s)	No. of steps	Time (min:s)	No. of steps
Average	09:10	185.5	08:01	95.5
Standard dev.	05:22	168.2	05:05	61.6

data). It is worthy to mention that there is no correlation between these measurements and the subjects' experience or data sets used in the tasks.

An interesting outcome from the analysis of the recorded logs was the patterns of interaction performed by the subjects. When using all resources of our framework, most users performed a very similar sequence of interaction steps. They first used boundary emphasis (three times in average) to obtain appealing thumbnails, which were often inspected in the second level, selected as parents and then used for combination or refinement of TFs by applying stochastic search (3.07 times in average). As a final step, the subjects applied the manual design tool to further refine the TF of a chosen thumbnail. We also noticed that experienced subjects made few level changes and passed to the final manual refinement phase only once. The dual domain interaction tool (clipping plane) was often used during manual design by all the subjects.

6. Conclusions and future work

Despite the considerable attention devoted to the TF specification problem, TF design is still a hard task. We are far away from an ideal solution, but several TF specification methods have proved to be useful. We developed an interactive general purpose volume visualization tool with high-quality volume rendering by adapting, extending and combining known TF specification techniques. Compared with other methods, ours successfully combines two different classes of approaches (image-driven and data-driven), allowing abstractions during the TF design process without preventing user control, since a friendly manual editing interface is also provided.

As a limitation, our method suffers from noise in volume data, and thus another straightforward future work is to apply a pre-processing step of filtering. This step could not only remove noise, but also enhance features present in the volume as shown by Fang et al. [19].

A particularly interesting future work is to extend our approach for multi-dimensional TF design. TF domains with more than two dimensions are very difficult to visually represent and explore, thus abstractions of TF representation are an attractive solution. With this in mind, image-driven approaches, like stochastic evolutive design, deserve special attention. Moreover, our two-level interaction interface is suitable for fast exploration of complex TF domains due to the simultaneous visualization of multiple possibilities. Following this assumption, we want to integrate our two-level interface and TF design tools with our proposal for multi-dimensional TF design [32], which is based on dimensional reduction.

Acknowledgments

We thank J.L. Prauchner and his co-authors for kindly providing us their tool, which was the starting point of ours, and the colleagues from the CG group of UFRGS that served as subjects in our evaluation. We also acknowledge the financial support from CNPq.

Table A1
Total number of interaction steps performed by the subjects in the Manual and Full tasks. For the Full task, the table shows the number of steps of each type: boundary emphasis, stochastic search, parent selection, level change and manual refinement

Subj.	Total		Number of individual interaction steps (Full task)				
	Manual	Full	B. Emph.	S. Search	Parent S.	Lev. Ch.	Man. R.
S1	160	37	2	2	9	7	17
S2	69	67	7	3	7	24	26
S3	212	73	1	2	5	9	56
S4	161	88	7	4	4	5	68
S5	31	34	3	1	8	5	17
S6	80	151	4	2	1	6	138
S7	485	142	8	7	4	38	85
S8	133	60	1	1	4	15	39
S9	621	280	1	1	2	16	260
S10	120	84	2	2	7	11	62
S11	326	42	3	3	9	9	18
S12	55	70	1	1	2	3	63
S13	149	110	1	0	0	3	106
S14	53	101	2	6	14	7	72
S15	128	108	1	8	12	12	75

Appendix A. Interaction steps during experiment

Table A1 shows the number of interaction steps performed by subjects during the visualization tasks in our experiment. One can see that, when using the complete framework (Full task), the number of interaction steps in manual refinement is strongly reduced for most subjects in relation to those measured during the Manual design task.

References

[1] Brodlie K, Wood J. Recent advances in volume visualization. Computer Graphics Forum 2001;20(2):125–48.
[2] Lorensen WE, Cline HE. Marching cubes: a high resolution 3d surface construction algorithm. In: SIGGRAPH '87: proceedings of the 14th annual conference on computer graphics and interactive techniques, vol. 21. New York, NY, USA: ACM Press; 1987. p. 163–9.
[3] Levoy M. Display of surfaces from volume data. IEEE Computer Graphics and Applications 1988;8(3):29–37.
[4] Kindlmann G, Durkin JW. Semi-automatic generation of transfer functions for direct volume rendering. In: Proceedings of IEEE symposium on volume visualization 1998. New York, NY, USA: ACM Press; 1998. p. 79–86.
[5] Lum EB, Ma K-L. Lighting transfer functions using gradient aligned sampling. In: Proceedings of IEEE Visualization 2004. Los Alamitos, CA, USA: IEEE Computer Society Press; 2004. p. 289–96.
[6] Tory M. A practical approach to spectral volume rendering. IEEE Transactions on Visualization and Computer Graphics 2005;11(2):207–16.
[7] Kniss J, Premoze S, Hansen C, Shirley P, McPherson A. A model for volume lighting and modeling. IEEE Transactions on Visualization and Computer Graphics 2003;9(2):150–62.
[8] Bruckner S, Gröller ME. Style transfer functions for illustrative volume rendering. Computer Graphics Forum 2007;26(3):715–24.
[9] Pfister H, Lorensen B, Bajaj C, Kindlmann G, Schroeder W, Avila LS, et al. The transfer function bake-off. IEEE Computer Graphics and Applications 2001;21(3):16–22.
[10] Kniss J, Kindlmann G, Hansen C. Multidimensional transfer functions for interactive volume rendering. IEEE Transactions on Visualization and Computer Graphics 2002;8(3):270–85.
[11] Kniss J, Kindlmann G, Hansen C. Interactive volume rendering using multidimensional transfer functions and direct manipulation widgets. In: Proceedings of IEEE visualization 2001. Los Alamitos, CA, USA: IEEE Computer Society Press; 2001. p. 255–62.
[12] Tzeng F-Y, Lum EB, Ma K-L. An intelligent system approach to higher-dimensional classification of volume data. IEEE Transactions on Visualization and Computer Graphics 2005;11(3):273–84.
[13] Kindlmann G, Whitaker R, Tasdizen T, Möller T. Curvature-based transfer functions for direct volume rendering: methods and applications. In: Proceedings of IEEE visualization 2003. Los Alamitos, CA, USA: IEEE Computer Society Press; 2003. p. 513–20.
[14] Hladuvka J, König A, Gröller E. Curvature-based transfer functions for direct volume rendering. In: Spring Conference on Computer Graphics 2000, vol. 16; 2000. p. 58–65.
[15] Tenginakai S, Lee J, Machiraju R. Salient iso-surface detection with model-independent statistical signatures. In: Proceedings of IEEE visualization 2001. Los Alamitos, CA, USA: IEEE Computer Society Press; 2001. p. 231–8.
[16] Kniss J, Premoze S, Ikits M, Lefohn A, Hansen C, Praun E. Gaussian transfer functions for multi-field volume visualization. In: Proceedings IEEE Visualization 2003. Los Alamitos, CA, USA: IEEE Computer Society Press; 2003. p. 497–504.
[17] Engel K, Kraus M, Ertl T. High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on graphics hardware 2001. New York, NY, USA: ACM Press; 2001. p. 9–16.
[18] Bajaj CL, Pascucci V, Schikore DR. The contour spectrum. In: Proceedings of IEEE visualization 1997. Los Alamitos, CA, USA: IEEE Computer Society Press; 1997. p. 167–73.
[19] Fang S, Biddlecome T, Tuceryan M. Image-based transfer function design for data exploration in volume visualization. In: Proceedings of IEEE visualization 1998. Los Alamitos, CA, USA: IEEE Computer Society Press; 1998. p. 319–26.
[20] Fujishiro I, Azuma T, Takeshima Y. Automating transfer function design for comprehensible volume rendering based on 3d field topology analysis (case study). In: Proceedings of IEEE visualization 1999. Los Alamitos, CA, USA: IEEE Computer Society Press; 1999. p. 467–70.
[21] He T, Hong L, Kaufman A, Pfister H. Generation of transfer functions with stochastic search techniques. In: Proceedings of IEEE visualization 1996. Los Alamitos, CA, USA: IEEE Computer Society Press; 1996. p. 227–36.
[22] Marks J, Andalman B, Beardsley PA, Freeman W, Gibson S, Hodgins J, et al. 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: ACM Press; 1997. p. 389–400.
[23] Pekar V, Wiemker R, Hempel D. Fast detection of meaningful isosurfaces for volume data visualization. In: Proceedings of IEEE visualization 2001. Los Alamitos, CA, USA: IEEE Computer Society Press; 2001. p. 223–30.
[24] Prauchner JL, Freitas CMDS, Comba JLD. Two-level interaction approach for transfer function specification. In: Proceedings of the 18th Brazilian symposium on computer graphics and image processing. Los Alamitos, CA, USA: IEEE Computer Society Press; 2005. p. 265–72.
[25] de M. Pinto F, Freitas CMDS. Two-level interaction transfer function design combining boundary emphasis, manual specification and evolutive generation. In: Proceedings of the 19th Brazilian symposium on computer graphics and image processing. Los Alamitos, CA, USA: IEEE Computer Society Press; 2006. p. 218–25.
[26] Dietrich CA, Nedel LP, Olabarriaga SD, Comba JLD, Zanchet DJ, da Silva AMM, de Souza Montero EF. Real-time interactive visualization and manipulation of the volumetric data using gpu-based methods. In: SPIE medical imaging 2004, vol. 5. San Diego, CA, USA; 2004. p. 181–92.
[27] König A, Gröller ME. Mastering transfer function specification by using Volumepro technology. Technical Report TR-186-2-00-07, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria; March 2000.
[28] Tzeng F-Y, Lum EB, Ma K-L. A novel interface for higher-dimensional classification of volume data. In: Proceedings of IEEE visualization 2003. Los Alamitos, CA, USA: IEEE Computer Society Press; 2003. p. 66–73.
[29] Rezk-Salama C, Keller M, Kohlmann P. High-level user interfaces for transfer function design with semantics. IEEE Transactions on Visualization and Computer Graphics 2006;12(5):1021–8.
[30] Šereda P, Bartroli AV, Gerritsen FA. Automating transfer function design for volume rendering using hierarchical clustering of material boundaries. In: Proceedings of eurographics/IEEE VGTC symposium on visualization 2006. Eurographics; 2006. p. 243–50.
[31] Šereda P, Bartroli AV, Serlie IWO, Gerritsen FA. Visualization of boundaries in volumetric data sets using LH histograms. IEEE Transactions on Visualization and Computer Graphics 2006;12(2):208–18.
[32] de M. Pinto F, Freitas CMDS. Design of multi-dimensional transfer functions using dimensional reduction. In: Proceedings of eurographics/IEEE VGTC symposium on visualization 2007. Eurographics; 2007. p. 131–8.
[33] Fujishiro I, Takeshima Y, Azuma T, Takahashi S. Volume data mining using 3d field topology analysis. IEEE Computer Graphics and Applications 2000;20(5):46–51.
[34] Bavoil L, Callahan SP, Scheidegger CE, Vo HT, Crossno P, Silva CT, et al. Vistrails: enabling interactive multiple-view visualizations. In: Proceedings of IEEE visualization 2005. Los Alamitos, CA: IEEE Computer Society Press; 2005. p. 135–42.
[35] ⟨http://www.opengl.org/resources/libraries/glut/⟩.
[36] Rademacher P. GLUI user interface library. ⟨http://glui.sourceforge.net/⟩.
[37] Shreiner D, Woo M, Jackie Neider T, Davis T, OpenGL Architecture Review Board. OpenGL programming guide: the official guide to learning openGL. 6th ed. Boston: Addison-Wesley; 2007.
[38] Fernando R, Kilgard MK. The Cg tutorial: the definitive guide to programmable real-time graphics. Boston: NVidia/Addison Wesley; 2003.