Custom Interface Elements for Improved Parameter Control in Volume Rendering

Marius Gavrilescu, Muhammad Muddassir Malik, and Eduard Gröller

***Abstract*— In volume visualization interfaces, rendering-related parameters are often manually editable through various controls and interface elements. Most of the time however, these offer little or no beforehand information on the resulting effects that would occur for certain parameter values or across the whole value domain. This makes parameter adjustment a trial and error process. We have developed techniques to anticipate these changes and display them on customized versions of popular interface elements, such as sliders or transfer function editors. Through the use of visualization means such as graphs, color mapping, and various other indicators, the influence of potential parameter changes on the volume rendering output can be assessed before any actual changes are made. This makes it easier for the potential user to work with such interfaces, while receiving feedback on parameter behavior and stability.**

I. Introduction

Volume visualization is a segment of computer graphics which deals with the exploration, classification and on-screen representation of information from three-dimensional, or often multi-dimensional datasets. Such datasets are typically acquired from scanning devices such as Computed Tomography (CT), Magnetic Resonance Imaging (MRI) or Rotational X-ray imaging. Volume visualization techniques are widely used in medical and industrial imaging, where a 3D representation of the available data is often more accessible, suggestive and visually appealing than traditional 2D grayscale slices, and may yield information otherwise hard to spot. Volume data is intuitively composed of atomic elements known as *voxels* (short for *volume pixel*), which are the equivalent in 3D space of the traditional pixels from 2D images. For the purposes of computer processing, voxel-based data is at first discretely sampled, then the samples are traversed and assigned optical properties, after which they are composited and projected into a 2D screen-space to produce an image. The previously mentioned steps are a loose outline of a generic volume rendering approach.

A typical volume rendering application consists of one or several viewports to display the images resulting from rendering the dataset. Multiple controls are used to manipulate the data processing. The complexity of these controls ranges from a simple slider to elaborate transfer function editing interfaces. While there exist numerous efforts to automate or semi-automate the visualization of this type of data [1], [2], [3], many volume rendering applications mostly leave it to the user to adjust the various parameters which control the on-screen outcome. Therefore, in many cases, an image which shows relevant information from a volume dataset is the result of parameter tweaking by means of sliders, interactive graphs, various widgets, and generally speaking, a variety of interface elements.

There is, however, a downside to allowing an extensive degree of manual control. Unless the user is very familiar with the particular dataset under analysis, the adjustment of parameters to obtain the desired results may prove to be a tedious and time-consuming trial-and-error task. Furthermore, while most volume rendering applications allow extensive control over the data, few if any relay feedback to the user as to how a hypothetical change in a parameter value might influence the resulting images. We attempt to reverse this situation through the development of interface elements which provide the user with a-priori knowledge into how a change in the interface control would reflect on the on-screen image. This would also aid in the assessment of parameter behavior and stability across its value-domain.

The paper is structured into several sections. After the introduction, we briefly outline volume rendering in general, and describe the rendering approach used in the paper. Section three deals with the metric used for comparing images resulting from sampled parameter values, and how this metric relates to the human vision system. In the following section, we present a couple of custom interface elements which allow control over their associated parameters, while at the same time automatically displaying information on parameter behavior and stability.

Manuscript revised on 31.08.2010. This work was supported in part by the "BRAIN - An Investment in Intelligence" doctoral Scholarship Program, within the Technical University of Iasi, Romania

The presented work has been partially supported by the Bridge-Project SmartCT and the K-Project ZPT (http://www.3dct.at) of the Austrian Research Promotion Agency (FFG).

Marius Gavrilescu is a PhD student from the Technical University of Iasi, Romania, and collaborates with the Vienna University of Technology, Austria within a joint PhD program (e-mail: mariusgv@cg.tuwien.ac.at, mariusgav42@yahoo.com).

Muhammad Muddassir Malik is with the School of Electrical Engineering and Computer Science, National University of Sciences and Technology, Pakistan (email: mmm@cg.tuwien.ac.at).

Eduard Gröller is associate professor at the Vienna University of Technology, Austria and adjunct professor of computer science at the University of Bergen, Norway (e-mail: groeller@cg.tuwien.ac.at).

We also present an on-screen approach for the dynamic visualization of parameter changes. We conclude by briefly emphasizing the significant aspects of the paper and by providing information on future work.

II. VOLUME RENDERING APPROACH

While a description of volume rendering techniques does not fall within the scope of this paper, we find it necessary to at least outline the basic methodology and point out the elements which are relevant to the contents of other sections.

The images found throughout the paper are produced using direct volume rendering (DVR). Unlike older techniques which employed "proxy geometry" [4] and triangle based surfaces to indirectly outline elements of interest from within the volume, DVR operates on the actual volume data, without the need to use additional geometric primitives. The dataset is typically uploaded into video memory as a 3D texture [5] and sampled discretely. A transfer function maps optical properties to the samples, which are then composited to form the desired image in screen-space. A popular algorithm which encompasses these steps is *ray casting* [4], which has the advantage of exploiting the hardware acceleration capabilities and the parallel architecture of modern graphics processing units (GPUs) [6], [7]. Fig. 1 shows an example of an image rendered via GPU ray casting. The corresponding transfer function is shown below the image.

The transfer function depicted in Fig. 1 maps opacity values to voxels according to their densities [8]. Regions of lower density, located toward the left of the graph in Fig. 1, have a very low opacity, which is reflected in the transparent appearance of the skin in the image rendered above the graph. Similarly, higher density regions such as bone are assigned a much higher opacity and are fully visible. Many volume rendering applications allow manual control over the shape of the transfer function. The control points in Fig. 1, marked with circles, are movable with the mouse and the in-between step-wise components of the function may be linear, cubic or may otherwise have any desired shape. The resulting image changes in real time to reflect the changes in the transfer function. The problem, as previously mentioned, is that such an interface offers little additional information on parameter effect. In other words, the users cannot know what the rendered image will look like if the transfer function is given a certain shape until they actually modify the transfer function. In Section 4, we present our method to address this issue.

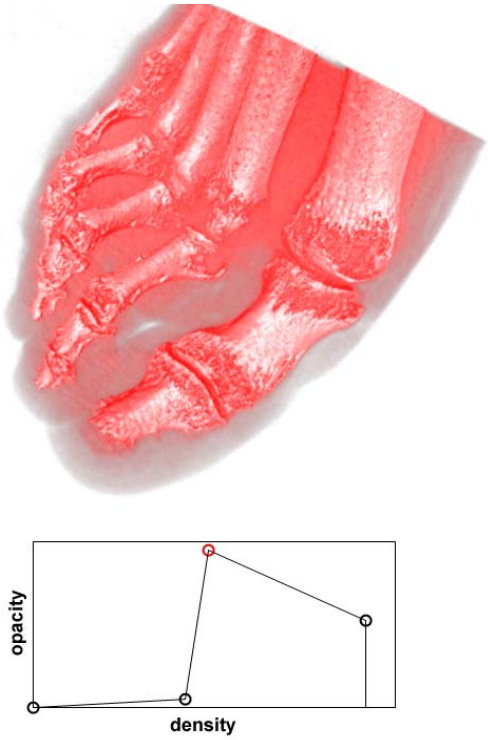

Fig. 1. Volume rendering of a medical dataset and the corresponding opacity transfer function

III. METRICS FOR IMAGE COMPARISON

Our approach to solving the problem of the lack of information in interface controls is mostly an image-oriented one. Given a particular parameter, we sample it across its value-domain and render an image for each sample. By comparing the resulting images we get an idea of how the parameter behaves across its domain, and how different values affect the on-screen outcome. This makes it possible to tailor popular interface controls to also show this parameter behavior, in addition to allowing control over its values. Image comparison is therefore an important piece of the puzzle and the choice in comparison metrics may significantly affect the outcome.

The metrics often involve a pixel-by-pixel comparison of the images, using some type of formula to assess the differences between color or intensity values, followed by an accumulation of these differences in a scalar. Therefore, for each pair of sampled parameter values, we end up with a scalar which shows the difference between using one sampled value versus using the other one. The domain of the parameter would then be characterized by an array of such scalars.

Among the most straightforward and computationally efficient metrics is the absolute mean difference, which essentially involves computing the mean of all pixel differences in the RGB color space, across the two compared images. Wilson et al. [9] provide an introduction

into some of the more common metrics, such as the root mean square (RMS), the signal to noise ratio, as well as their own Δ_g metric, based on the Sobolev norm and the Hausdorff metric. Chan et al. [10] have developed an image comparison method based on the Canny edge detection algorithm. However, such metrics are mathematically defined around pixel differences. They do not take into account aspects pertaining to subjective human perception, though they may incidentally correlate with the human vision system. Furthermore, it is difficult to assess the robustness and efficiency of such metrics, since it is the user who has to relate the information obtained from applying the metrics to actual on-screen changes.

A more straightforward approach to defining a suitable metric is to design bottom-up. We design a metric around the human vision model, considering aspects of the human perception of color and intensity. Efforts in this direction have been made and are well documented in literature [11], [12], [13]. Such metrics could be described as *perception-based image comparison metrics*. They take into account factors such as hue angle, color distance, pixel placement or an estimated viewer distance, among others. As is often the case, there is also a trade-off between the complexity of the metric and the processing speed. Perception-based metrics are typically computationally intensive. As the complexity of the metric increases, performance becomes a significant problem, and the processing of complex metrics at high resolutions and for hundreds of parameter values may take hours even on a relatively powerful machine.

Considering the above, we have developed a metric which takes into account some of the previously mentioned perceptual aspects, while attempting to provide an efficient means of image comparison. The metric is processed in the following steps:

- the general area of the volume in screen space is isolated from the rest of the image, since the background presents no relevant information

- the behavior of the human eye, which only looks at a few details at a time as opposed to the image as a whole, is approximated by analyzing a finite number of random rectangular sub-regions within the image. These regions are then weighed according to their size and color uniformity. This is based on an approach proposed by Matkovic [14].

- a noise removal filter is applied, since changes in noise have little impact on the perceived change in the image. In other words, noise, even when it changes significantly, is still perceived as noise.

- for the purpose of assessing pixel differences, we change to the CIE-Lab color space [15], which is perceptually closer to the human interpretation of color than the RGB model.

- the accumulated difference for corresponding rectangular regions between each pair of images is calculated using the color distance ΔE^*_{ab} [15].

- finally, the amount of variation between each pair of images is computed as the weighted mean of the values from the rectangular regions.

The accuracy and relevance of the metric are difficult to assess, since they are, to a significant degree, subjective matters. Performance-wise, migrating some of the previously mentioned steps to the GPU has shown improvement, though still insufficient for use in real time. However, the trade-off between complexity and speed has so far proven satisfactory on commercial hardware.

IV. Custom Interface Controls

For demonstrative purposes, we consider two basic parameters involved in volume rendering: the step size used when sampling the volume during ray casting and a basic transfer function control which adjusts the threshold of an isosurface and its opacity.

The information regarding the behavior of these parameters across their domain is incorporated into common interface elements by means of information visualization techniques.

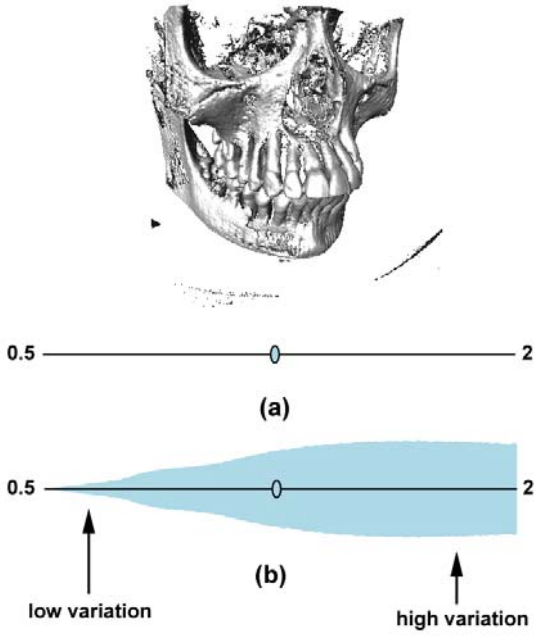

Fig. 2. Rendered dataset with traditional and custom sliders which control the step size. The custom slider also depicts information on parameter behavior

Slider widgets are frequently used for the adjustment of one-dimensional parameters such as the step size. However, a basic slider does not provide any a-priori information regarding the changes that would happen upon

changing the position of the pointer (Fig. 2a). Unless the user is very familiar with the volume under consideration, moving the slider to get the desired result is a matter of trial and error. The slider in Fig. 2b shows the magnitude of change which occurs in the rendered image when the pointer is moved to a neighboring location.

The slider may be further customized as needed, as seen in Fig. 3. Often it might be necessary to more closely inspect a region of the slider where significant changes occur. For this purpose, the specific region can be selected (Fig. 3a) and non-linearly scaled to fill a larger portion of the slider (Fig. 3b). It can take up the entire available length. The borders of the regions keep their initial values, but there is more space along the length of the slider thus allowing more precision in selecting a desired position for the pointer. This process of fine tuning is useful for portions of the slider where there is an abrupt variation in the amount of parameter effect.

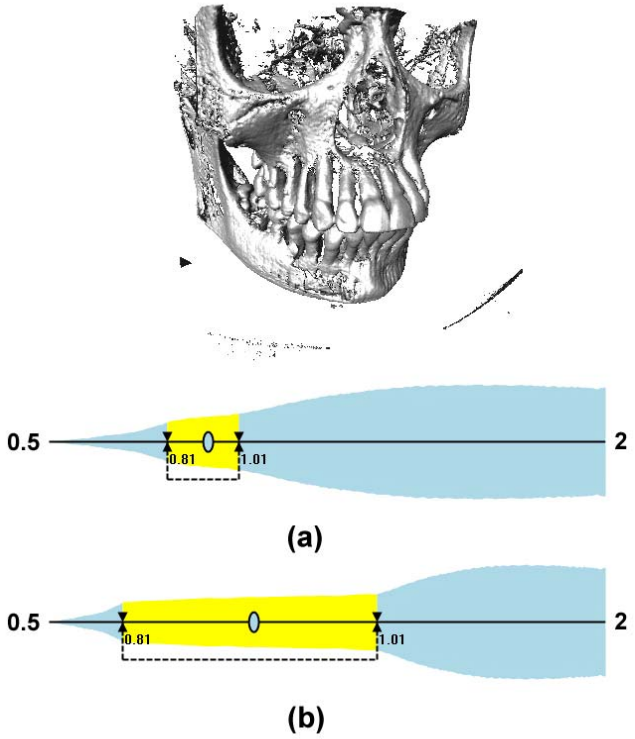

Fig. 3. The selection (a) and non-linear scaling (b) of a slider region for fine tuning purposes

The technique may be fully automated by partitioning the slider into small regions and assigning each region a portion of the slider of a length proportional to the variation taking place within the region. We thus end up with a *perceptually uniform* slider, where the perceived changes in the rendered image are proportional to the distance by which the pointer was moved along the slider.

Performance-wise, we took 300 samples from the domain of the parameter, to generate values for the graph of the slider. We rendered and compared the images obtained for each pair of sampled parameter values. The computations for processing the custom slider and rendering the volume in Fig. 2 took approximately 16.2 seconds on our test machine, an Intel Core I7 with 6 GB of RAM and a GeForce GTX 280 GPU.

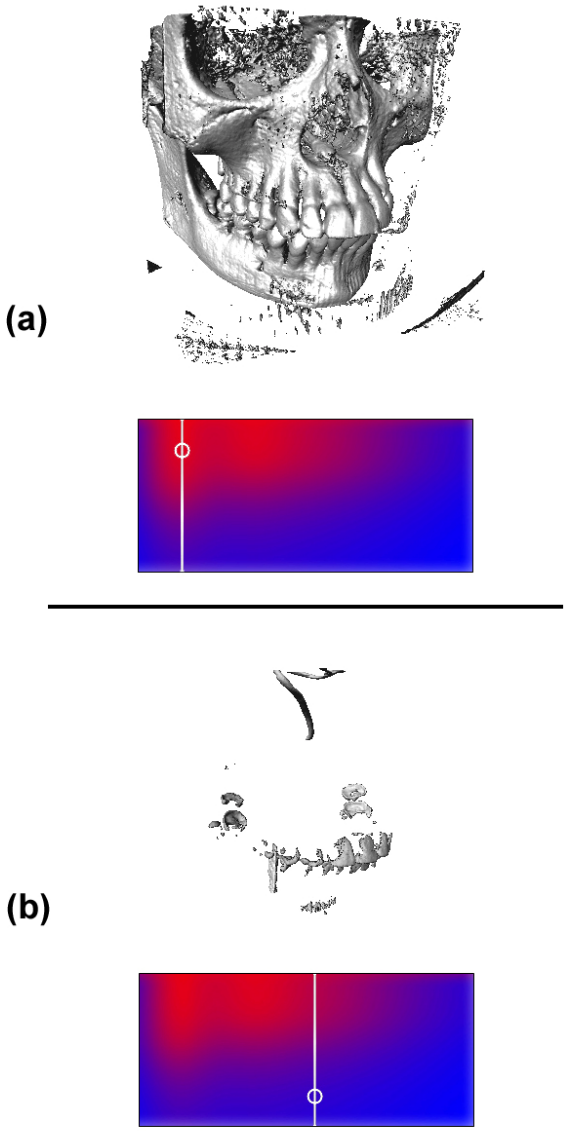

Fig. 4. Custom isosurface control where the magnitude of change is color-coded. (a) and (b) show two positions of the pointer at opposite ends of a region of transition which signifies a variation in parameter stability

The concept of parameter behavior displayed on a custom version of a frequently used interface element also extends to transfer function controls. For this purpose, we consider a simplified version of the editable graph depicted in Fig. 1. The simplified graph allows the adjustment of an isosurface threshold when the pointer is moved horizontally, and the adjustment of the opacity of the isosurface, when the pointer is moved vertically. As

previously, the parameters are sampled and the corresponding images are rendered and compared.

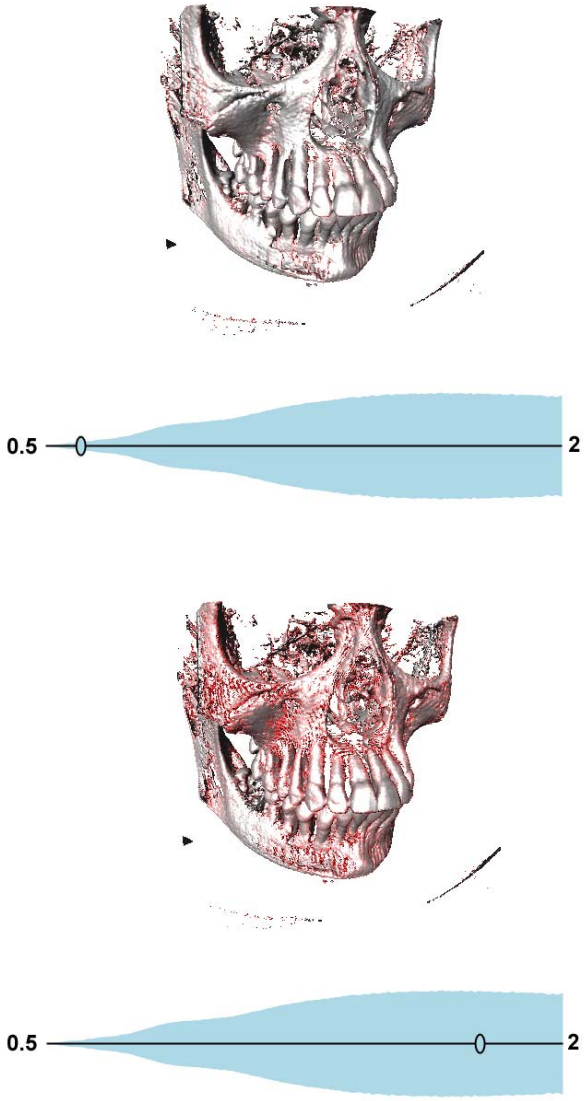

Fig. 5. Changes illustrated on the rendering of the volume for two positions on the step size slider

Fig. 4a and 4b show an isosurface control where the influence of parameter changes is color coded through a red-blue color range. Red areas correspond to positions of the pointer which cause significant changes in the rendered image (shown above the control), whereas blue areas denote regions where the change in parameter values has little effect on the outcome. The purple transition area in-between the red and blue is a region of varying stability, whereby the parameters gain stability as the pointer is moved from the red region toward the blue one.

We took 50 samples on the horizontal axis and 20 on the vertical axis, for a total of 1000, to generate values for color coding. Our test machine took around 49 seconds to process the isosurface control from Fig. 4.

Changes occurring when manipulating an interface control can be shown on the rendered volume itself, as depicted in Fig. 5. The changes are shown using a red coloration of varying intensity. When the pointer is in the low variation region on the slider, little changes are visible. If the pointer is moved to an area showing greater variations, this is reflected in the more frequent intensely red regions on the volume.

However, this approach has the downside of being intrusive and inflicting possibly unwanted changes in the final resulting image. The direct color encoding of image variation may obstruct desired information when superimposed on the volume rendering. One way to avoid this is to use a *viewing lens*, i.e. to restrict the display of changes to a bordered circular region which the user can freely move using the mouse, as illustrated in Fig. 6.

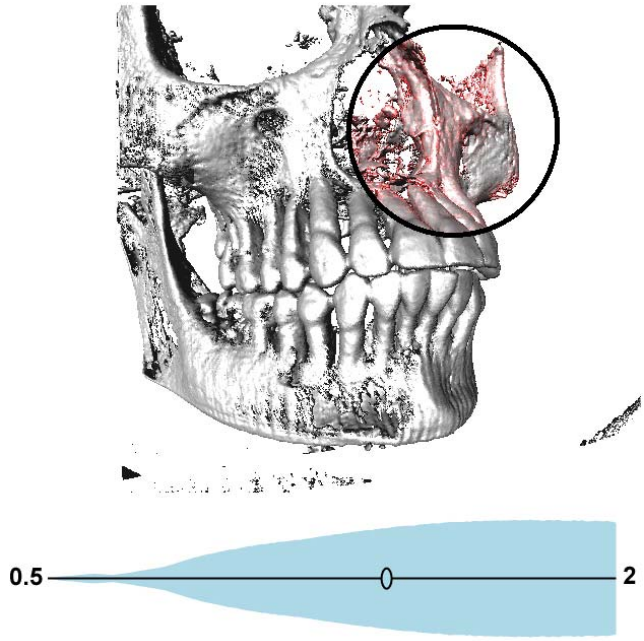

Fig. 6. The viewing lens restricts the display of changes to a mouse-controlled circular region

V. CONCLUSIONS

Interface elements customized to offer information on parameter behavior are meant to make parameter adjustment more efficient and straightforward. The approaches described thus far do not provide an exhaustive analysis of the parameters involved in volume rendering. They do offer a-priori information on the variations which would manifest upon certain changes in the discussed parameters, which may aid in making volume exploration and related interface elements easier to interact with.

There is naturally a lot of room for improvement and extension in this area, mostly in the direction of expanding

the mentioned techniques to more complex contexts, while at the same time working on new information visualization methods for the purposes of parameter analysis. Future development in this direction includes the extension of the concept of customized interface elements to single and multidimensional transfer function controls, while focusing on other potentially more relevant parameters and improving the metrics used for image comparison.

ACKNOWLEDGMENT

Marius Gavrilescu thanks the members of the Vis-Group from the Institute of Computer Graphics and Algorithms of the Vienna University of Technology for their invaluable support and help with research. He would also like to thank Professor Vasile Manta from the Technical University of Iasi, Romania, for his help and supervision.

REFERENCES

[1] J. Zhou, M. Takasuka, "Automatic transfer function generation using contour tree controlled residue flow model and color harmonics", *IEEE Trans. Vis. Comput. Gr.*, vol. 15, no. 6, pp.1482-1488, 2009.

[2] P. Kohlmann, S. Bruckner, A. Kanitsar, and E. Gröller, "LiveSync: Deformed Viewing Spheres for Knowledge-Based Navigation", *IEEE Trans. Vis. Comput. Gr.*, vol. 13, no 6, pp. 1544-1551, October 2007.

[3] F. F. Bernardon, L. K. Ha, S. P. Callahan, J. L. D. Comba, and C.T. Silva, "Interactive Transfer Function Specification for Direct Volume Rendering of Disparate Volumes," SCI Institute Technical Report, No. UUSCI-2007-007, University of Utah, 2007.

[4] M. Hadwiger, J. M. Kniss, C. Rezk-Salama, D. Weiskopf, and K. Engel, *Real-time Volume Graphics*. Wellesley, MA: A K Peters, 2006.

[5] O. Wilson, A. Van Gelder, and J. Wilhelms, "Direct volume rendering via 3D textures", University of California, Santa Cruz, CA, Tech. Rep. UCSC-CRL-94-19, 1994.

[6] A. Kratz, M. Hadwiger, R. Splechtna, A. Fuhrmann, and K. Bühler, "High quality volume rendering of medical data for virtual environments", in *Proc. Int. Conf. Computer Aided Surgery Around the Head (CAS-H 2007)*, Innsbruck, Austria, 2007, pp. 83-85.

[7] I. Viola, A. Kanitsar, and E. Gröller, "GPU-based frequency domain volume rendering", in *Proc. Spring Conference on Computer Graphics (SCCG 2004)*, Budmerice, Slovakia, 2004, pp. 49-58.

[8] G. Kindlmann, "Transfer functions in direct volume rendering: Design, interface, interaction", in *SIGGRAPH 2002 Course Notes*, 2002.

[9] D. L. Wilson, A. J. Baddeley, and R. A. Owens, "A new metric for grey-scale image comparison", *Int. J. Comput. Vision*, vol. 24, no. 1, pp. 5-17, 1997.

[10] W. C. Chan, M. V. Le, and P. D. Le, "A wavelet and Canny based image comparison", in *Proc. 2004 IEEE Conf. Cybernetics and Intelligent Systems*, Singapore, pp. 329-333.

[11] M. Pedersen, and J. Y. Hardeberg, "A New Spatial Hue Angle Metric for Perceptual Image Difference", in *Computational Color Imaging: Second International Workshop (CCIW 2009)*, Saint-Etienne, France, 2009, pp 81-90.

[12] S. I. Titov, "Perceptually Based Image Comparison Method", in *Proc. Int. Conf. Graphicon 2000*, Moscow, Russia, 2000, pp 8-16.

[13] J. P. Farrugia, S. Albin, and B. Peroche, "A perceptual adaptive metric for computer graphics", in *Proc. WSGC2004*, Plzen, Czech Republic, 2004, pp. 49-52.

[14] K. Matcovic, "Tone mapping techniques and color image difference in global illumination", Ph.D. dissertation, Vienna University of Technology, Austria, 1997.

[15] Y. Ohno, "CIE fundamentals for color measurements", in *Proc. IS&T NIP16 International Conference on Digital Printing Technologies*, Vancouver, Canada, 2000, pp. 540-545.