# New technique for transfer function specification in direct volume rendering using real-time visual feedback

Charl P. Botha and Frits H. Post
Computer Graphics and CAD/CAM, Delft University of Technology, The Netherlands

## ABSTRACT

Direct volume rendering (DVR) is a very useful visualisation technique. However, the difficulty in specifying a suitable transfer function often discourages its application to visualisation problems. This work presents a technique for providing meaningful and fast visual feedback that greatly facilitates the transfer function specification process. The feedback is based on a slice-based preview of the actual volume rendering that can be calculated in real-time and can be super-imposed on a slice-based view of the data that is being rendered.

**Keywords:** volume rendering, visualisation, transfer function specification

## 1. INTRODUCTION

Direct volume rendering[1,2] (DVR) is an important and useful technique for visualising large 3D data-sets in medicine and science. One problem that hinders effective use of DVR is the difficulty of specifying a transfer function[3] for a specific data-set. The transfer function assigns optical properties, such as opacity and colour, to the data values in the data-set. A good transfer function can make a vast difference in rendered image quality, but it is difficult to derive such a function automatically as it is very dependent on the semantics of each specific data-set.

This work presents a new and practical scheme for the interactive specification and optimisation of a transfer function that offers some very desirable characteristics. The user is assumed to understand the significance of the data, but does not necessarily need any visualisation experience. The technique makes use of this domain-specific knowledge by giving visual feedback about the current state of the transfer function in terms of its correlation with the data-set. The user is guided in her modifications of the transfer function by how closely the provided visual feedback matches her comprehension and expectation of the data-set that she wishes to make a volume rendering of. In other words, the focus is moved away from transfer function specification and towards emphasising the display of anatomical features, such as blood vessels, tissues or organ shapes.

Section 2 discusses related work and section 3 describes the feedback technique itself. In section 4 we document an implementation of the algorithm in our medical visualisation platform and section 5 discusses the output. Section 6 summarises the work and mentions possible future work.

## 2. RELATED WORK

The existing schemes for transfer function specification range from the completely manual to the completely automatic.

The most common scheme is the trial-and-error method which mostly involves manually editing the transfer function and checking the resultant volume rendering at regular intervals. If real-time direct volume rendering hardware or software is available *that can cope with a continually changing transfer function*, the volume rendering can be updated continuously. If such facilities are not available, this method can be very laborious and time-consuming. Even if they are available, relating structures in the volume rendering to their recognisable counter-parts in the traditional grey-scaled slice view can be problematic. This method requires a high level of visualisation background from the user.

Author e-mail: {c.p.botha,f.h.post}@its.tudelft.nl web: http://visualisation.tudelft.nl/.

A method that tries to alleviate the reliance on the user's visualisation expertise involves creating hundreds of sample renderings corresponding to a range of automatically generated transfer functions. The user picks the renderings that are most satisfactory and in this way implicitly selects the most suitable transfer functions. This technique is based on the design galleries idea.[4] The major challenge is to generate random transfer functions that will result in the widest spread of dissimilar output renderings.[5] Because possibly hundreds of volume renderings have to be made for the user to choose from, this scheme also requires real-time volume rendering functionality to be truly useful as an interactive technique.

Bajaj's contour spectrum[6] consists of scalar data and contour attributes that are calculated over the scalar values in a data-set. It is displayed as a collection of "signature curves", each curve representing a different attribute. These curves assist the user in selecting iso-values for surface extraction and consequent visualisation and are therefore also most useful when assisting the user in creating a transfer function.

Kindlmann's semi-automatic transfer function generation method[7] makes the reasonable assumption that the features of interest in a data-set are the boundaries between different materials. By making use of the relationship between the data-values and their first and second derivatives along the gradient direction Kindlmann's method can generate transfer functions that makes the boundaries between different materials visible in the resultant volume rendering.

The trial-and-error and the design galleries schemes function in a feedback-based fashion. Both of them require real-time direct volume rendering facilities to reach their full potential. Neither addresses the problem of yielding feedback on the *fidelity* of a volume rendering resulting from a specific transfer function. Displaying the volume rendering resulting from a particular transfer function yields information about the expected optical composition but contains almost no explicit information about the correlation between the rendering and the data that is being rendered. This makes it difficult to quantify the impact of small changes as they are being made to the transfer function. In addition, it is entirely possible that a transfer function is created that results in a volume rendering that appears satisfactory but is a mis-representation of the structures present in the data-set.

Our method attempts to remedy these shortcomings by providing explicit feedback on the correlation between volume-rendered structures and structures in the data-set that is being rendered. It does not aim to replace the semi-automatic schemes, but can be used to augment them.

## 3. METHOD

A common way of examining 3D data-sets, especially in medical applications, is to view the greyscale representation of a voxel-thick slice of the data. The position and orientation of the slice can be changed interactively so that the user has the impression of "moving" through the data. Clinical users are cognitively well-adapted to recognise structures in data-sets that are represented in this fashion.

In addition to allowing the user to examine the data in this fashion, she is also granted access to the transfer function and is able to manipulate it directly by adjusting five piecewise linear functions representing the hue, saturation, value, scalar opacity and gradient opacity transfer function components. Our method is not constrained to this particular type of transfer function interaction. Far simpler interaction schemes can also be employed.

Throughout this interaction, all voxels in the current slice are transformed with the current transfer function and a new slice is created by applying straight-forward mappings on the transformed voxels. Henceforth, this new slice will also be referred to as the overlay slice. The mapped result, $M = \langle R_m, G_m, B_m, A_m \rangle$, is a function of the transformed voxel optical properties, i.e. $M = f(\langle R_t, G_t, B_t, A_t \rangle)$.

The new slice is super-imposed on a greyscale representation of the current slice by alpha-blending. If the greyscale representation is $X = \langle R_g, G_g, B_g, A_g \rangle$, the result of the alpha-blending is, as per usual:

$$(1 - A_m)X + A_m M = \langle (1 - A_m)R_g + A_m R_m, (1 - A_m)G_g + A_m G_m, (1 - A_m)B_g + A_m B_m, A_m \rangle$$

Several different mappings can be used, each accentuating some aspect of the transfer function in the feedback process:

- With $M = \langle A_t, 0, 0, A_t \rangle$, only the opacity is visualised, but both as red intensity and as blending opacity. At high opacities $A_t$, the red overlay is brighter and also obscures most of the greyscale slice view with which it is blended.

- Another way of visualising only the opacity is by making use of $M = \langle 0, 1.0, 0, A_t \rangle$. The intensity of the green overlay is kept at a maximum, but the opacity is visualised by the amount of the greyscale slice that is visible through the new super-imposed slice. This mapping yields better results than the previous one at low opacities due to the fact that the green intensity is not affected by opacity.

- Both the colour and opacity are visualised with $M = \langle R_t * A_t, G_t * A_t, B_t * A_t, A_t \rangle$. The colour is scaled with opacity, so that the opacity is fed back as both blending transparency and colour intensity. Remember that the colour of a voxel is scaled with the opacity during raycasting. However, because our scheme does not accumulate optical properties as is the case in volume rendering, this results in low opacities decreasing and potentially neutralising colour intensities.

- With
$$M = \langle R_t * A_t, G_t * A_t, B_t * A_t, 1 - e^{\frac{-A_t}{\tau}}) \rangle \quad (1)$$
we try to emulate what happens with low opacities in a raycasting. Recall that the absorption term in the direct volume rendering integral is
$$I(D) = I_0 e^{-\int_0^D \alpha(t)dt}$$
where $I(D)$ is the light that reaches the eye (via the light-attenuating volume), $I_0$ is light that is present at the opposite side of the volume, and $\alpha(t)$ is the instantaneous opacity.[8, 9] Thus, during raycasting, low opacities accumulate very rapidly whereas opacities that are closer to unity accumulate more slowly. The last component in equation 1 emulates this behaviour. The time-constant $\tau$ determines how rapidly the component increases with increasing $A_t$.

- The mapping $M = \langle 0, 1.0, 0, 1 - e^{\frac{-A_t}{\tau}}) \rangle$, combines the maximum intensity monochromatic overlay slice with the low opacity compensation. This is a very good way of working with low opacities, as they are compensated for *and* they are not allowed to affect colour intensity.

- By ignoring the scaling effect of opacity on the colour intensity and by compensating for low opacities, $M = \langle R_t, G_t, B_t, 1 - e^{\frac{-A_t}{\tau}}) \rangle$ yields the best all-round results and most closely approximates the resultant volume rendering. This mapping will be referred to as the "unscaled colour and compensated opacity mapping".

Once super-imposed, the new slice clearly shows the expected visibility and optical properties of structures in the final volume rendering. Because the preview is voxel-registered with the data, the one-to-one correspondence between the greyscale representation of the raw data and volume rendered structures is clearly visualised and the fidelity of the transfer function can be checked.

As the user makes small changes in the transfer function, the effects of these small changes are immediately reflected in the preview. In effect, a "soft" segmentation is performed on the volume data with the specified transfer function in real-time. When the super-imposed segmentation agrees with the user's comprehension of the structures in the data, the transfer function has been implicitly optimised and the resulting volume rendering will also reflect the user's expectations.

## 4. IMPLEMENTATION

The Delft Shoulder Computer Assisted Surgery (DSCAS1) platform[10] was used to create a test implementation of the feedback technique. DSCAS1 offers a rich application programming interface and flexible structure for the implementation of visualisation algorithms.

Its design is based on the concept of object pools and accompanying pool managers that act as brokers for the objects that their pools contain. Access to objects in a specific pool is only granted via the relevant pool manager. In our case, there is a data entity pool, a pipeline pool and a view pool. A data entity is a passive object that carries information, e.g. a medical dataset or a polygonal surface description. A pipeline is an active element that can perform in-place processing on data-entities or create new ones. Views are used to interact with data entities and pipelines.

In addition to this, there is a communication infrastructure based on a detached variant of the observer design pattern, i.e. views or pipelines can subscribe via the relevant pool manager to be observers of any object. Destroyed observers are automatically detected and removed from the observer lists.

This design makes it possible to keep code units separate and independent. In addition, each object that is implemented as part of a more complex algorithm can be re-used in subsequent implementations without additional work.

The feedback technique is implemented as a pipeline and makes use of existing DICOM,[11] HDF[12] or raw data entities for volume input. It registers as an observer of an existing transfer function data entity and of the slice-viewer that is being used to view the volume data. It creates a slice of output volume data that represents the new slice that is to be super-imposed. The slice-viewer is then attached to this output data as well, thus resulting in the alpha-blending of the volume data and the overlay.

Whenever the user changes the current slice in the slice-viewer or modifies the transfer function the feedback pipeline is notified and subsequently recomputes the overlay slice according to the currently selected mapping. The user interacts with the transfer function via a transfer function view. The standard transfer function editor is depicted in figure 1. Other interaction methods can easily be implemented as transfer function views. New feedback mappings can be selected by attaching a feedback pipeline *view* to the pipeline. The user immediately sees the result of her changes in the slice-viewer that she is interacting with.

When the transfer function yields a satisfactory overlay slice, it can be connected to a raycasting pipeline to create a direct volume rendering of the volume data. The raycasting pipeline can also be connected at the beginning of the transfer function specification process so that the user can monitor both an interactive lower-resolution raycasting and the overlay generated by the technique whilst optimising the transfer function. In addition, the overlay slice can be viewed both overlaid on a greyscale represenation of the raw data or separately.

## 5. RESULTS

In this section, we show previews and resulting volume renderings of the "tooth" and "heart" datasets used in the Transfer Function Bake-Off.[3] In order to appreciate the effectiveness of the scheme, one needs to interact with it and, very importantly, view the previews in colour. We show only the unscaled colour with compensated opacity mapping (the last mapping listed in section 3) as this is still illustrative when grey-scaled.

For the opacity compensation, we have selected time-constant $\tau = 0.25$. This yields good results with all data-sets that we tested on.

In all the data-sets we have tested with, we were able to converge on a suitable transfer function in a few minutes by making use of only the overlaid preview computed by our feedback scheme.

Figure 2 shows a preview and resultant volume rendering of the "tooth" dataset and figure 3 shows a preview and resultant volume rendering of the "heart" dataset.

## 6. CONCLUSIONS

We have presented a straight-forward scheme to provide meaningful and fast feedback during the transfer function specification process. The feedback consists of a slice-based preview that is alpha-blended with a grey-scaled slice of the data that is being volume rendered. The preview is calculated in real-time while the user interacts with the transfer function and current slice position. Importantly, the alpha-blended preview supplies information about the expected optical characteristics as well as the *fidelity* – due to the voxel-registration of the generated previews – of the resultant volume rendering.

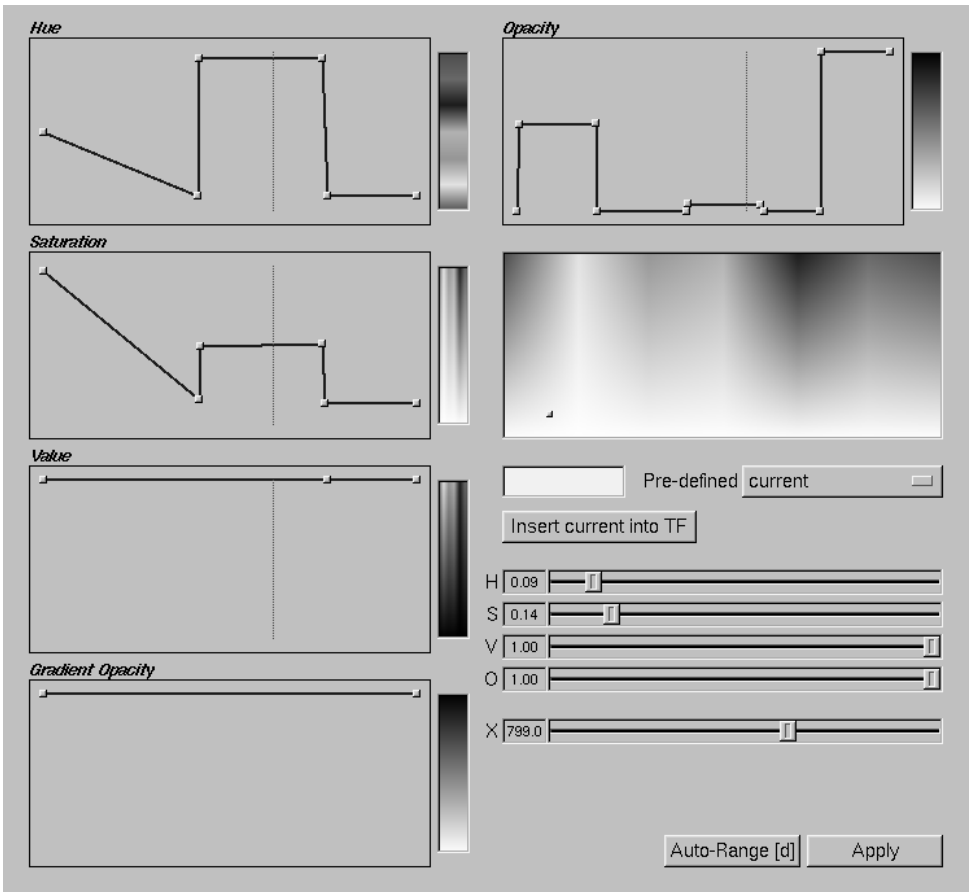


**Figure 1.** The standard transfer function editor allows the user to interact with an attached transfer function. In this case, we have chosen for piecewise-linear hue, saturation, value, scalar opacity and gradient opacity functions. An arbitrary number of points can be added, removed and moved in any of the functions. The feedback technique can obviously be used with more complex transfer function representations.


**Figure 2.** Preview (with unscaled colour and compensated opacity mapping) and resulting volume rendering of "tooth" data-set.


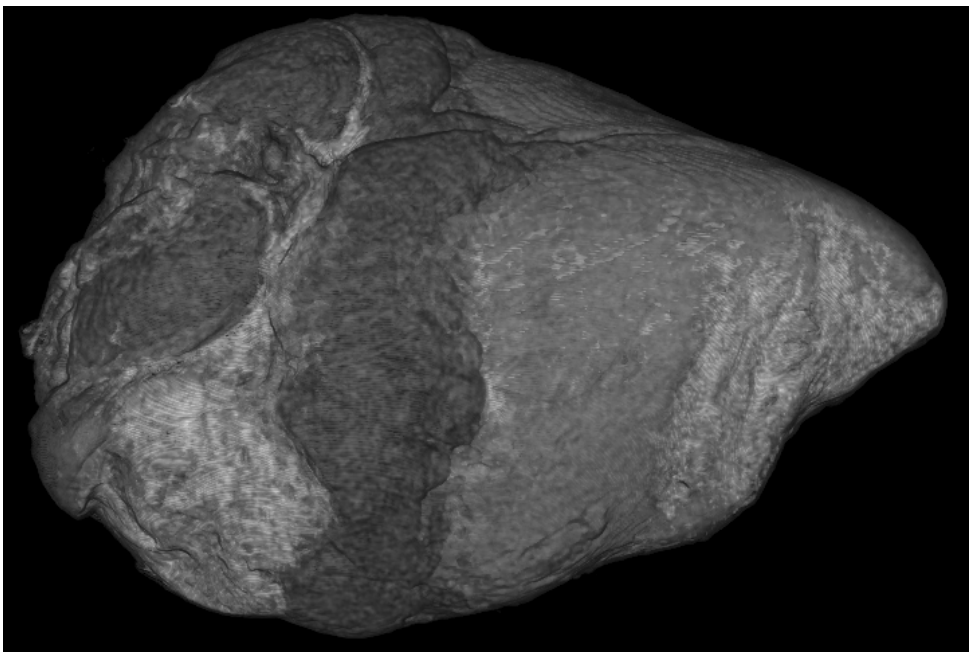

**Figure 3.** Preview (with unscaled colour and compensated opacity mapping) and resulting volume rendering of "heart" data-set.

This scheme is simple to implement on commonly available platforms and does not require specialised DVR hardware. It makes effective use of the user's knowledge of the data and therefore does not rely on any specific data-model or metric. Because of the easily understandable real-time data-registered visual feedback, an applicable transfer function can be very rapidly optimised with minimal user effort.

Our experiments with opacity compensation have opened up further opportunities for research. We wish to investigate the possibility of doing more accurate colour and opacity composition in real-time by simplifying the raycasting process and performing necessary pre-processing. This will enable the scheme to generate previews that are closer in appearance to the final volume renderings.

## ACKNOWLEDGMENTS

This research is part of the DIPEX (Development of Improved endo-Prostheses for the upper EXtremities) program of the Delft Interfaculty Research Center on Medical Engineering (DIOC-9).

## REFERENCES

1. M. Levoy, "Display of surfaces from volume data," *IEEE Computer Graphics and Applications* **8**, pp. 29–37, May 1988.
2. M. Meissner, J. Huang, D. Bartz, K. Mueller, and R. Crawfis, "A Practical Evaluation of Popular Volume Rendering Algorithms," in *Proc. Volume Visualization and Graphics Symposium*, pp. 81–90, 2000.
3. H. Pfister, B. Lorensen, C. Bajaj, G. Kindlmann, W. Schroeder, L. Avila, K. Raghu, R. Machiraju, and J. Lee, "The transfer function bake-off," *IEEE Computer Graphics and Applications* **21**, pp. 16–22, Jan-Feb 2001.
4. J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber, "Design galleries: a general approach to setting parameters for computer graphics and animation," in *ACM SIGGRAPH Conference on Computer Graphics*, pp. 389–400, 1997.
5. T. He, L. Hong, A. Kaufman, and H. Pfister, "Generation of transfer functions with stochastic search techniques," in *Visualization '96*, pp. 227–234, 489, IEEE, 1996.
6. C.Bajaj, V.Pascucci, and D.Schikore, "The Contour Spectrum," in *Visualization '97*, pp. 167–173, IEEE, 1997.
7. G. Kindlmann and J. W. Durkin, "Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering," in *IEEE Symposium on Volume Visualization*, pp. 79–86, 1998.
8. N. Max, "Optical models for direct volume rendering," *IEEE Transactions on Visualization and Computer Graphics* **1**, pp. 99–108, June 1995.
9. W. Krueger, "The application of transport theory to visualization of 3D scalar data fields," in *Proceedings of Visualization '90*, pp. 273–280, 481–2, IEEE, 1990.
10. C. P. Botha and F. H. Post, "A Visualisation Platform for Shoulder Replacement Surgery," in *Interactive Medical Image Visualization and Analysis (IMIVA) - Satellite Workshop of MICCAI 2001*, S. D. Olabarriaga, W. J. Niessen, and F. Gerritsen, eds., pp. 61–64, October 2001.
11. National Electrical Manufacturers' Association, "NEMA Standard PS3: Digital Imaging and Communications in Medicine (DICOM)," 2000.
12. NCSA University of Illinois at Urbana-Champaign, *HDF Reference Manual*, December 2000. `ftp://ftp.ncsa.uiuc.edu/HDF/HDF/Documentation/HDF4.1r4/Ref_Manual/`.