Fusing Features in Direct Volume Rendered Images

Yingcai Wu, Huamin Qu, Hong Zhou, and Ming-Yuen Chan

Department of Computer Science and Engineering
Hong Kong University of Science and Technology
{wuyc, huamin, zhouhong, pazuchan}@cs.ust.hk

Abstract. In this paper, we propose a novel framework which can fuse multiple user selected features in different direct volume rendered images into a comprehensive image according to users' preference. The framework relies on three techniques, i.e., *user voting*, *genetic algorithm*, and *image similarity*. In this framework, we transform the fusing problem to an optimization problem with a novel energy function which is based on user voting and image similarity. The optimization problem can then be solved by the genetic algorithm. Experimental results on some real volume data demonstrate the effectiveness of our framework.

1 Introduction

Direct volume rendering (DVR) is a powerful and flexible volume visualization tool and has been widely used in many fields. However, to effectively and intuitively explore volumetric data through DVR still remains a challenging issue. Due to the data occlusion and human perception, one of the difficulties is to develop effective visualization techniques capable of revealing multiple features simultaneously. To address this issue, many approaches like illustrative visualization and importance-based techniques have been proposed. However, most techniques have more or less limitations. On the other hand, as it is easier for users to reveal a specific feature than to highlight multiple features simultaneously in a *direct volume rendered image* (DVRI), a feasible solution to the problem may allow users to select multiple features in distinct DVRIs, and then automatically fuse those features into a comprehensive DVRI. Therefore, users can focus on one feature each time and the task for exploring multiple features is simplified.

We assume that a series of DVRIs have been generated and cached along with the *transfer functions* (TFs) used. To fuse multiple features in different DVRIs, there are two straightforward solutions, i.e., to blend those images, or to generate a new DVRI with a new TF created by linearly combining the previous TFs. However, they both fail to achieve our goal in most cases. Image blending does not work for our purpose due to its limitations like losing depth cues and introducing misleading information [1]. Linear combination of the TFs does not work either. This can be mainly attributed to the nonlinear operations of the integration used in DVR. For example, given two DVRIs and their corresponding TFs (see Figure 2(a) and 2(b)), if users want to reveal features appearing in both 2(a) and 2(b) by linear combination of the TFs, they can only obtain something like 2(c) no matter what α and β are, which is far from what they expect such as 2(d). Because there are other features, such as the intestines and muscles between intensity d_1 and d_2 , linear combination in this example does not work.

G. Bebis et al. (Eds.): ISVC 2006, LNCS 4291, pp. 273–282, 2006.

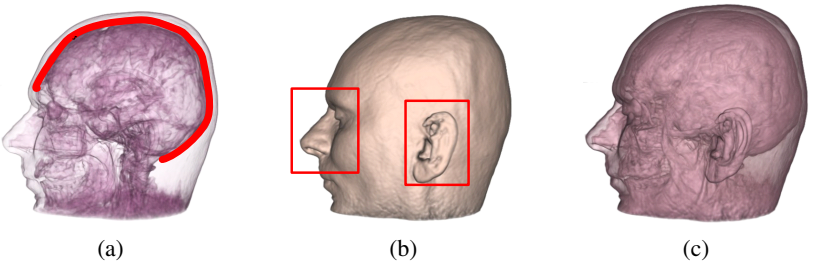


Fig. 1. Experiment on an MRI human head dataset: (a)-(b) Parent images with user selected features; (c) Child image generated by fusing the user selected features shown in (a) and (b)

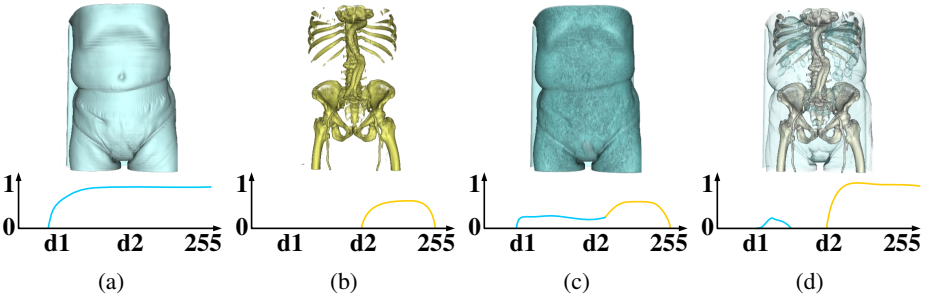


Fig. 2. (a)-(b) Parent images 1 and 2, and their TFs TF_1 and TF_2; (c) Child image rendered with linearly combined TF : $TF3 = \alpha \times TF1 + \beta \times TF2$, where $\alpha = 0.3$ and $\beta = 1$; (d) Child image rendered with our method by fusing (a) and (b)

To solve the general feature fusing problem, we propose a novel fusing framework, which relies on three techniques, i.e., *user voting*, *genetic algorithm (GA)*, and *image similarity*. In this framework, we transform the fusing problem to an optimization problem with a novel energy function based on user voting and image similarity, which can then be solved by the GA. User votes includes the user selected features in DVRIs and the corresponding user given scores representing how much the users like the features. A selected feature could be either a region of interest (with high scores) or the context (with low scores). Our approach contains some advanced characteristics. First, as it is easier for users to select features in DVRIs than in volume slices [2], our approach allows users to select multiple features directly in DVRIs instead of in volume slices. The features can be then automatically fused together. Second, we develop a general framework for fusing multiple user-selected features in DVRIs. Compared with previous methods such as image blending and TF interpolation, our method is more robust and general. Third, as a semi-automatic method, our approach integrates user knowledge into the fusing process. Since the visualization goal highly depends on the tasks and users, our system can gain valuable input from users by user voting.

This paper is organized as follows: After reviewing previous work in Section 2, we give an overview of our system in Section 3. The solution encoder/decoder is explained

in Section 4. A genetic algorithm used to solve the optimization problem is described in Section 5. In Section 6, we provide the details of the image similarity metric. Experimental results and discussions are presented in Section 7. We conclude and suggest some future work in Section 8.

2 Related Work

Feature-based Visualization. Weiskopf et al. [3] proposed several interactive clipping methods by exploiting the powerful capability of GPU. Viola et al. [4] developed a novel importance-based approach capable of enhancing important features while retaining necessary context by generating cut-away and ghosted images from volumetric data. Volume illustration techniques, first introduced by Ebert et al. [5], provide an alternative way for focus + context visualization with abstraction techniques (non-photorealistic rendering). Wang et al. [6] presented an interactive GPU-assisted volume lens to magnify the regions of interest while preserving the context by compressing the other volume regions.

Transfer Function (TF) Design. An excellent survey on TF design can be found in [7]. Marks et al. [8] developed an image-centric system which automatically generates many perceptually different images and organizes them efficiently for users to select. Kindlemann and Durkin [9] presented a histogram data structure used to semi-automatically obtain a good TF with users' guidance. The images generated by the both approaches can be used as inputs to our system. Ma [10] proposed a novel approach based on image graphs to facilitate visual data exploration. König and Gröller [11] developed a TF design interface paradigm which provides several specification tools for each search domain. Although Ma's and König et al.'s methods used linear combination of TFs, they cannot always get the expected results (see Figure 2).

Genetic Algorithms. Genetic algorithms are widely used in many fields like computer graphics [12]. He et al. [13] first employed GAs to generate TFs. Our approach is inspired by their work, but aims at fusing different features rather than TF design from scratch. In addition, compared with He et al.'s approach, our approach is based on image similarity and user knowledge (or user voting). Users are allowed to control which features will be retained or enhanced in the child images by user voting. House et al. [14] used a GA to choose visualization parameters to optimize visualization quality.

3 System Overview

To simplify the presentation, we first introduce the following terms: *Parent image* for the image with user selected features for fusing; *Child image* for the fused image from parent images; *Parent TF* for the TF corresponding to the parent image; *Child TF* for the TF corresponding to the child image. All the images are rendered by DVR.

Our system consists of four components as shown in Figure 3, i.e., *solution encoder/decoder*, *genetic algorithm solver*, *direct volume renderer*, and *image similarity evaluator*. The encoder/decoder component specifies the genome representation by analyzing the parent transfer functions (TFs) (denoted as $TF_1 \cdots TF_n$ in Fig. 3), and then

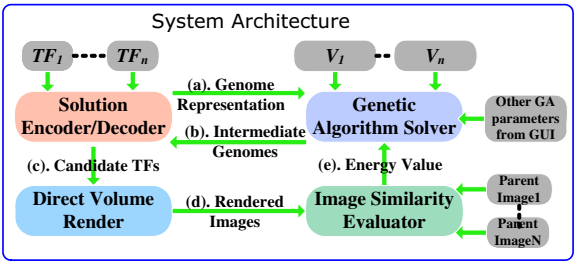


Fig. 3. System Architecture

sends the genome representation to the genetic algorithm (GA) solver (see Figure 3 (a)). The genetic algorithm solver generates new genomes (the encoded TFs) by imitating the process of natural evolution. The expected similarity values denoted as $V_1 \cdots V_n$ in Fig. 3 and the calculated similarity values received from the image similarity evaluator for each parent image are used to form an energy function which measures the differences between the real calculated similarity values and the expected similarity values. The smaller the energy value, the fitter the corresponding genome. Each intermediate genome created in the course of the GA evolution is then sent back to the solution encoder/decoder (see Figure 3(b)). The decoder converts the genome to a candidate TF which is then passed to the direct volume renderer (see Figure 3(c)). The direct volume renderer generates a child image using DVR techniques for the candidate TF. The generated image is then passed to the image similarity evaluator (see Figure 3(d)). It measures the similarity between the child image and each of the parent images. The image similarity values are then sent back to the genetic algorithm solver (see Figure 3(e)) to begin the next cycle of refinement.

4 Solution Encoder/Decoder

The solution encoder/decoder specifies the genome representation by analyzing the parent TFs. To define the optimal genome representations [15], our approach represents a TF as a one dimensional (1D) array of floating numbers. The component first smoothens the parent TFs using a gaussian function to filter out the high frequencies to obtain bandlimited signals. After that, it samples TFs adaptively above the Nyquist frequency. The samples are then used to specify the genome representation. In addition, they can be used to restrict the search space to improve the GA performance. For example, in Figure 4 there are two parent TFs denoted by different line patterns to be fused. The points on the axis of the scalar value are the union of the sampling positions of the parent TFs. The vertical dashed lines start with 0 and end with the maximum opacity value of the parent TFs. They act as the range of the opacity values of the corresponding points on the axis of the scalar value. All the opacity values on the forementioned points (on the axis of the scalar value) of each candidate child TF constitute a genome.

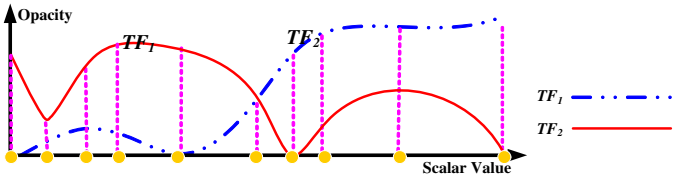


Fig. 4. The genome representation (the points on the axis of the scalar value) obtained by analyzing parent transfer functions TF_1 and TF_2

5 Genetic Algorithm

A *Genetic Algorithm* (GA) is a search algorithm imitating the process of natural evolution [15]. It is particularly useful for searching solutions to optimization problems, especially when the search space is huge and unknown. The pseudo-code for the simple genetic algorithm used in our system is shown as follows:

```
1: Randomly create an initial population of n genomes (encoded TFs)
2: repeat
3:     repeat
4:         Select a pair of genomes from the current population using roulette wheel scheme as
           parent genomes such that fitter (or better) genomes are more likely to be chosen
5:         Crossover (two-point crossover) the selected pair with probability p_r or exactly copy
           (or clone) the pair with probability 1 − p_r to form two new genomes
6:         Mutate the two newly created genomes with mutation probability p_m
7:     until n new genomes (offsprings) have been created
8:     Replace the current population with the n new genomes
9: until Terminating conditions such as the converging of the GA are met
```

Energy functions are used to measure the fitness of genomes, and return the measurement to the GA where the energy values are used to determine which genomes in the current population are more likely to be selected to survive. Thus, the energy function has a great impact on the performance of GAs. For our system, we exploit an energy function based on image similarity and user voting to objectively evaluate the fitness of genomes as follows:

$$F = \sum_{k=1}^{n} V_k * |V_k - S_k| \tag{1}$$

Where n is the number of parent images, and V_k represents the vote (or the scores) given by users for the features in parent image k, and S_k denotes the computed image similarity value between the child image and parent image k. The more detailed definition of S_k can be found in Equations (2) in Section 6.

The V_k, from another point of view, can be also considered as the similarity value expected by users between the child image and parent image k. It ranges from 0 to 1. It penalizes the difference between the computed similarity S_k and the user-voted (or user-expected similarity) value V_k. It can guarantee that the child image will get proportional

contributions from all the parent images and will not be overwhelmed by any single one. The GA used in our system views the genomes with smaller energy values as the fitter ones for the problem. Using the energy function, we are able to transform the feature fusing problem to an optimization problem, i.e., minimizing the energy function.

6 Image Similarity

In our system, we employ a contour-based similarity metric to compare two DVRIs. The contour is one of the most important perception clues to 3D structures and can be used to compute the similarity of objects. Thus, our system first converts the DVRIs to grey-scale images, and detects the edge images from the grey-scale images using the *Canny edge detector*. Because most contours appearing in the DVRIs also appear in the grey-scale images, our contour-based metric still works well. After that, a Gaussian filter is applied to smooth the edge images so that the pixels without an edge covering can also obtain contributions from the nearby edges. We set the size of the filter to be 5×5 for 512×512 images, and 3×3 for 256×256 images. The image similarity can then be computed pixel by pixel.

The image similarity value S_k between the edge image of the child image and the edge image of each parent image k is computed as follows:

$$S_k = \frac{\sum_{y=1}^{height} \sum_{x=1}^{width} Q(x,y)}{N_{parent}} \tag{2}$$

$$Q(x,y) = \begin{cases} 1 \text{ if } P(x,y) < threshold \\ 0 \quad Otherwise \end{cases} \tag{3}$$

$$P(x,y) = |K_{(x,y)} - K'_{(x,y)}| \tag{4}$$

where N_{parent} in Equation (2) is the number of all pixels on the edges of the parent k's edge image. *threshold* in Equation (3) is a parameter set by the system, and $P(x,y)$ in Equation (4) represents the difference between two corresponding pixels and K and K' are the Gaussian filtered child and parent edge images with resolution (*width*, *height*).

Notice that we consider only the pixels on the parent edge image k for S_k, i.e., Equation (2) is only computed if $K'_{(x,y)} \neq 0$. If there are user-selected features in the DVRIs which are to be compared, our system considers only the pixels within these features. Thus, the N_{parent} in Equation (2) becomes the number of pixels within the features on the edges of the parent edge image.

To determine the default *threshold* in Equation (3), we conducted extensive experiments on real volume data. All the edge images to be compared are 8-bit grey-scale images. After extensive experiments, we observed that the similarity values returned by the similarity evaluator was very similar to what users perceived when the *threshold* ranged from 60 to 90. Based on this, the *threshold* was fixed to be 80 for all experiments in this paper. An example of computed tomography (CT) human head volume data is shown in Figure 5. Figure 5(a) and 5(c) are two parent images. Figure 5(e) was generated by fusing features appearing in Figure 5(a) and 5(c). Figure 5(b), 5(d), and 5(f) are their corresponding edge images. Figure 5(g) shows the similarity value distribution in terms of different *threshold*. From the figure, we can observe that when the *threshold*

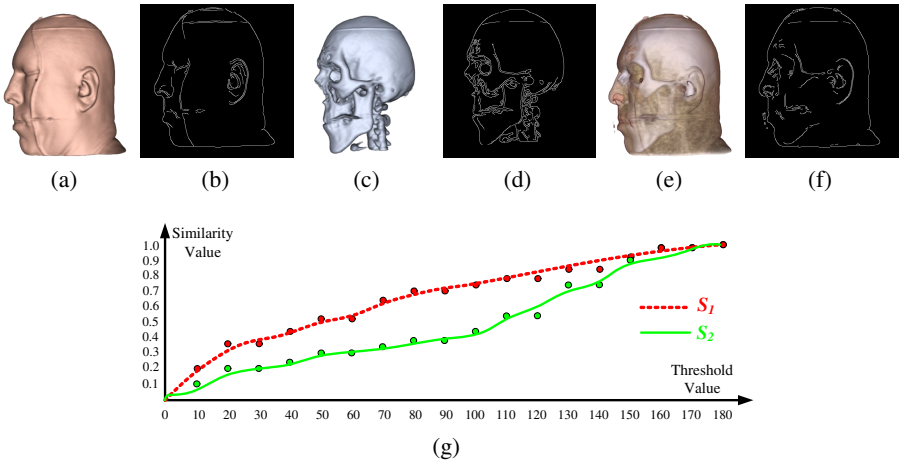


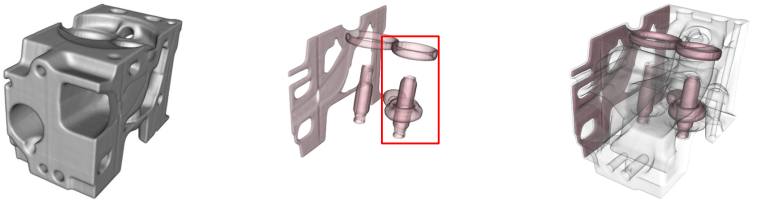
Fig. 5. Experiments for image similarity on CT human head volume data with different *threshold* in Equation 3: (a) Parent image 1; (b) Parent edge image 1; (c) Parent image 2; (d) Parent edge image 2; (e) Child image; (f) Child edge image; (g) Similarity value distribution in terms of different *threshold*

ranges from 60 to 90, the similarity value S_1 for Figure 5(e) and 5(a) will vary from 0.53 to 0.7, while S_2 for Figure 5(e) and 5(c) will vary from 0.3 to 0.38. This is similar to what users perceive. We also carried out the experiments on other volumetric data, such as CT engine data, CT human tooth data, and Magnetic resonance imaging (MRI) head data and found similar results. The image resolution was 512×512 and the size of the gaussian filter we applied was 5×5. Finally, we did the above experiments on images with 256×256 image resolution and with a 3×3 gaussian filter and obtained similar findings as well. Thus, to improve the system's performance, we can use the 256×256 image resolution and 3×3 gaussian filter in the fusing process for DVR and image similarity computing.

7 Experiment Results and Discussions

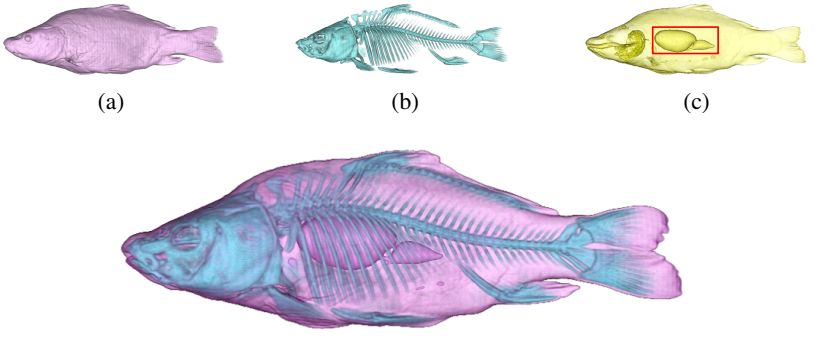
Our system was implemented in C++ based on the VTK 5.0 library [16]. We tested the system on a Pentium(R) 4 3.2GHz PC with 1GB RAM and an Nvidia Geforce 6800 Ultra GPU with 256MB RAM. The sampling rate of DVR was two samples per voxel along each ray. For the sake of performance, the rendered image resolution was 256×256 in the fusing process and was then switched to 512×512 after fusing. Following Jong's suggestion [17], we set the population size to be 10, the crossover rate to be 0.6, the mutation rate to be 0.05, n to be 20, and k to be 1.005 for the GA. In the following experiments, we obtained sufficiently good results within acceptable time frames.

We carried out the first experiment on a CT engine dataset ($256 \times 256 \times 128$) to verify the effectiveness of the fusing while taking user-selected features into account. We gave 0.4 scores to all the features shown in Figure 6(a) and 0.6 scores to the feature selected by a rectangle in Figure 6(b). The result is shown in Figure 6(c). From the result, we can



(a) (b) (c)

Fig. 6. Experiment on a CT engine dataset: (a) Parent image 1; (b) Parent image 2 with a feature selected with a rectangle; (c) Child image generated with $V_1 = 0.4$ and $V_2 = 0.6$



(a) (b) (c) (d)

Fig. 7. Experiment on a CT carp dataset: (a) Parent image 1; (b) Parent image 2; (c) Parent image 3 with a user-selected feature indicated by a rectangle; (d) Child image obtained by fusing features in (a) with 0.3 scores, features in (b) with 0.4 scores, and the user-selected feature in (c) with 0.3 scores

see that the user-selected feature is emphasized in the child image according to users' preference. The result was generated within 48 seconds.

Our approach was not limited to fusing only two DVRIs. In the second experiment, we fused the features in multiple DVRIs into a comprehensive one based on users' preference. We carried out the experiment on a CT carp dataset ($256 \times 256 \times 512$). In this experiment, we fused all the features shown in Figure 7(a) with 0.3 scores and 7(b) with 0.4 scores, and a feature (the swimming bladder) selected by a rectangle in Figure 7(c) with 0.3 scores together into a comprehensive image. Figure 7(d) shows the resulting image generated within 40 seconds.

We did the last experiment on an MRI head dataset ($256 \times 256 \times 256$) to demonstrate that our approach is able to generate a fused image containing the user selected features with clearer contours. The image shown in Figure 1(b) was created by a semi-automatic TF design method [9]. However, the semi-automatic method could not easily obtain an image revealing the inner structures of the head. Thus, we manually created an image (see Figure 1(a)) with a simple TF capable of revealing the inner structures of the data. But this image was not as good as what we expected, because the ear disappeared and

the inner structures like the brain were not clear enough. To solve these problems, we specified the features with curves in Figure 1(a) and with rectangles in 1(b). After that, we fused these features with $V_1 = 0.6$ and $V_2 = 0.4$. In the fusing process, our approach favored only those candidate images with clear contours within the specified regions. Figure 1(c) shows the result generated within 47 seconds.

8 Conclusion and Future Work

In this paper, we present a novel visualization technique which fuses multiple user selected features in distinct DVRIs into a comprehensive DVRI. We develop a general framework for the fusing problem. We view the fusing problem as a parameter optimization problem, which can then be solved using a genetic algorithm. To measure the fitness of a candidate image, we propose an energy function based on image similarity and user voting. Our approach is easy to use, especially for non-experts like physicians, because it is much easier for them to extract one feature in the volume data than to reveal multiple features simultaneously. In addition, our system allows users to directly select and identify features in DVRIs, which is more convenient and accurate than in volume slices. To fuse multiple features from several DVRIs, users usually do not need to set any parameters other than the scores voted for the features.

There are many possible venues for future work. Our implementation was not highly optimized for performance. The running time for each fusing on the tested volumetric datasets was about 50 seconds. We plan to exploit GPU-accelerated GAs [18] and DVR to greatly reduce the running time of our algorithm. We also plan to develop more sophisticated image similarity metrics for complex transparent DVRIs and test them with more real volume datasets. We assume that the viewpoint stays unchanged in the fusing process. We will extend our system to fuse features in DVRIs rendered from different viewpoints.

Acknowledgements

This work was supported by RGC grant CERG 618705 and HKUST grant DAG 04/05 EG02. We would like to thank the anonymous reviewers for their helpful comments.

References

1. Baudisch, P., Gutwin, C.: Multiblending: displaying overlapping windows simultaneously without the drawbacks of alpha blending. In: Proceedings of the SIGCHI conference on Human factors in computing systems. (2004) 367–374
2. Tzeng, F.Y., Lum, E.B., Ma, K.L.: An intelligent system approach to higher-dimensional classification of volume data. IEEE Transactions on Visualization and Computer Graphics **11** (2005) 273–284
3. Weiskopf, D., Engel, K., Ertl, T.: Interactive clipping techniques for texture-based volume visualization and volume shading. IEEE Transactions on Visualization and Computer Graphics **9** (2003) 298–312

4. Viola, I., Kanitsar, A., Gröller, M.E.: Importance-driven feature enhancement in volume visualization. IEEE Transactions on Visualization and Computer Graphics **11** (2005) 408–418
5. Rheingans, P., Ebert, D.: Volume illustration: Nonphotorealistic rendering of volume models. IEEE Transactions on Visualization and Computer Graphics **7** (2001) 253–264
6. Wang, L., Zhao, Y., Mueller, K., Kaufman, A.E.: The magic volume lens: An interactive focus+context technique for volume rendering. In: Proceedings of IEEE Visualization. (2005) 367–374
7. Kindlmann, G.: Transfer functions in direct volume rendering: Design, interface, interaction. In: Course notes of ACM SIGGRAPH. (2002)
8. Marks, J., Andalman, B., Beardsley, P.A., Freeman, W., Gibson, S., Hodgins, J., Kang, T., Mirtich, B., Pfister, H., Ruml, W., Ryall, K., Seims, J., Shieber, S.: Design galleries: a general approach to setting parameters for computer graphics and animation. In: Proceedings of ACM SIGGRAPH. (1997) 389–400
9. Kindlmann, G., Durkin, J.W.: Semi-automatic generation of transfer functions for direct volume rendering. In: Proceedings of Volume Visualization Symposium. (1998) 79–86
10. Ma, K.L.: Image graps - a novel interface for visual data exploration. In: Proceedings of IEEE Visualization. (1999) 81–88
11. König, A.H., Gröller, E.M.: Mastering transfer function specification by using volumepro technology. In: Proceedings of Spring Conference on Computer Graphics. (2001) 279–286
12. Sims, K.: Artificial evolution for computer graphics. In: Proceedings of ACM SIGGRAPH. (1991) 319–328
13. He, T., Hong, L., Kaufman, A., Pfister, H.: Generation of transfer functions with stochastic search techniques. In: Proceedings of IEEE Visualization. (1996) 227–234
14. House, D., Bair, A., Ware, C.: On the optimization of visualizations of complex phenomena. In: Proceedings of IEEE Visualization. (2005) 87–94
15. Mitchell, M. In: An Introduction to Genetic Algorithms. MIT Press (1996)
16. Kitware: (The visualization toolkit. www.vtk.org)
17. Jong, K.A.D.: An analysis of the behavior of a class of genetic adaptive systems. PhD thesis (1975)
18. Wong, M.L., Wong, T.T., Fok, K.L.: Parallel evolutionary algorithms on graphics processing unit. In: Proceedings of IEEE Congress on Evolutionary Computation. (2005) 2286–2293