Interactive Transfer Function Design Based on Editing Direct Volume Rendered Images

Yingcai Wu, *Student Member*, *IEEE*, and Huamin Qu, *Member*, *IEEE*

Abstract—Direct volume rendered images (DVRIs) have been widely used to reveal structures in volumetric data. However, DVRIs generated by many volume visualization techniques can only partially satisfy users' demands. In this paper, we propose a framework for editing DVRIs, which can also be used for interactive transfer function (TF) design. Our approach allows users to fuse multiple features in distinct DVRIs into a comprehensive one, to blend two DVRIs, and/or to delete features in a DVRI. We further present how these editing operations can generate smooth animations for focus + context visualization. Experimental results on some real volumetric data demonstrate the effectiveness of our method.

Index Terms—Direct volume rendering, image editing, intuitive user interface, morphing, transfer function design, focus+context, animation.

1 Introduction

Volume Visualization helps people gain insight into volumetric data using interactive graphics and imaging techniques. The data from real applications such as medical imaging and computational fluid dynamics often contain multiple complex structures. Due to the occlusion of 3D objects, revealing all these structures and presenting their 3D relationships in one image is very challenging. A widely used technique to address this problem is *Direct Volume Rendering* (DVR). By assigning different transparency values to the voxels of volumetric data via *transfer functions* (TFs) and then compositing them into *Direct Volume Rendered Images* (DVRIs), DVR can reveal more information than traditional surface-based rendering. However, the effectiveness of DVR largely depends on the TF used. Appropriate TFs allow users to reveal important features in the data, whereas inappropriate ones may obscure these features. Finding appropriate TFs is difficult in practice. One major reason is that the search space for finding TFs is huge even for one-dimensional (1D) TFs, not to mention multidimensional TFs. In most cases, finding a good TF becomes a tedious and time consuming trial-and-error process. Some excellent automatic or semiautomatic approaches have been developed, but this issue is far from being completely resolved [32].

For the end users of visualization systems, such as physicians, who may not have much experience in volume rendering and TF design, a user-friendly approach that allows them to intuitively explore volumetric data is very desirable. Physicians usually prefer to directly work on 2D slice images rather than in the TF domain, as this is how they are trained in medical schools. Thus, it is more straightforward for them to identify features in 2D slice images. On the other hand, some 3D structures can be more easily identified in DVRIs than in 2D gray-scale slice images. Therefore, it is more intuitive and convenient for users to directly work on DVRIs. Usually, a number of partially good DVRIs can be easily generated by previous volume visualization methods [17], [29]. These DVRIs may only partially satisfy a users' demands. For example, some DVRIs may contain too much context that should be removed, and some features that appear in different DVRIs should be combined into one comprehensive image to reveal their spatial relationships. Based on this observation, some DVRI editing operations such as fusing features from different DVRIs, blending two DVRIs, and erasing unwanted features in DVRIs, may be very useful in practice.

Editing DVRIs is not a trivial problem. There are two straightforward solutions: to use traditional 2D image editing operations or to generate new DVRIs by linearly combining several TFs. However, they both fail to achieve our goals in most cases. The traditional 2D image editing operations do not work for our purpose. Compared with traditional images, DVRIs have some special features. Traditional images usually show objects in a real-world setting, thus opaque surfaces are often presented, whereas DVRIs are used to reveal information contained in 3D volume data so multilayer transparent surfaces are usually displayed. In some DVRIs containing fuzzy structures, there are even no clear surfaces. Therefore, we may get some artificial or physically incorrect images instead of DVRIs and some unexpected artifacts may be introduced if we directly apply traditional editing operations to DVRIs. For example, if we directly blend two DVRIs by alpha blending, the resulting image may miss depth cues and introduce misleading information [1]. The linear combination of the TFs may not work either. This would mainly be attributed to the nonlinear operations of the integration used in DVR. For example, given two DVRIs and their corresponding TFs (see Figs. 1a and 1b), if users want to reveal features appearing in both Figs. 1a and 1b by the linear combination of the TFs, they can only obtain something like

• *The authors are with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. E-mail: {wuyc, huamin}@cse.ust.hk.*

Manuscript received 12 Sept. 2006; revised 14 Jan. 2007; accepted 23 Feb. 2007; published online 23 Mar. 2007.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-0163-0906.
Digital Object Identifier no. 10.1109/TVCG.2007.1051.

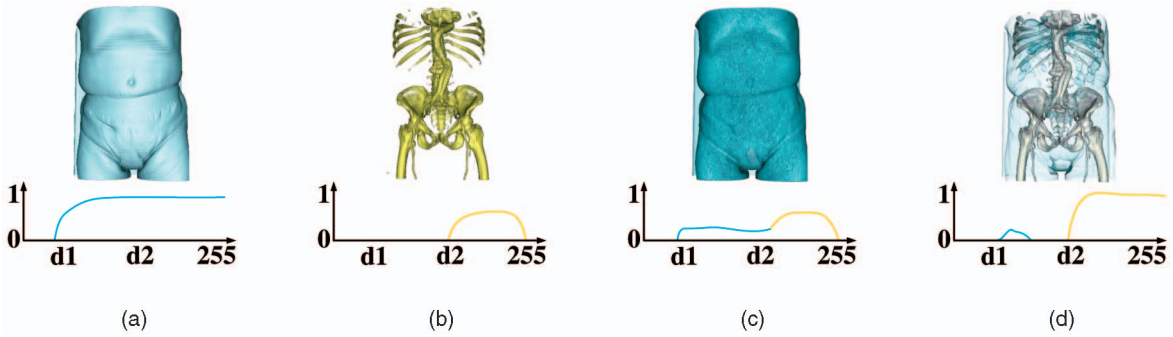

Fig. 1. (a) and (b) Source images 1 and 2 and their TFs, TF_1 and TF_2. (c) DVRI rendered with a linearly combined TF: $TF3 = \alpha \times TF1 + \beta \times TF2$, where $\alpha = 0.3$ and $\beta = 1$. (d) DVRI rendered with our method by fusing (a) and (b).

Fig. 1c no matter what α and β are, which is far from what they expect as in Fig. 1d. As there are other features, such as the intestines and muscles between intensity d_1 and d_2 (see the TFs in Fig. 1), the linear combination in this example does not work.

In this paper, we propose a general and robust framework for users to directly edit DVRIs. We focus on images generated by full DVR, as they are more general and complex. Thus, we do not consider other DVRIs such as MIP and x-ray style images in this paper. The system first transforms the DVRI editing problem to an optimization problem with an energy function created for each editing operation. The energy function is based on the image similarity values between the target DVRI and the source DVRIs. Then, a stochastic search algorithm is used to search for the TF that can generate a desirable DVRI by minimizing the energy function. *Genetic Algorithms* (GAs) are particularly effective for searching out good solutions when the search space is huge and poorly understood and, therefore, are utilized to solve the optimization problem. As contour is one of the most important perception clues for 3D structures, our system exploits a contour-based approach to objectively evaluate the image similarity values.

The uses of our framework are twofold: as an image editing tool and as an interactive TF design method. For users without expertise in TF, our system is a Photoshop-style editing tool for DVRIs, whereas TFs are only used internally and will not be exposed to users. For expert users, our system can show the generated TF, which can be further edited or manually fine-tuned by users. As a TF design method, our system allows users to interactively and intuitively design TFs from simple to complex by gradually editing simple DVRIs into comprehensive ones.

Our approach contains some advanced characteristics. To the best of our knowledge, we are the first to propose a framework for directly editing features in DVRIs. In contrast to the traditional 2D image operations, our system can produce DVRIs without losing depth cues and introducing incorrect information. Compared with the linear combination of TFs, our approach is more general and robust. As DVRIs contain more information and can better reveal 3D structures than 2D image slices, directly editing DVRIs is more intuitive and convenient for users than the traditional operations on 2D image slices [40]. With the framework, we are able to circumvent the time-consuming and nonintuitive TF design and manipulation process by allowing users to fuse multiple features in distinct DVRIs into a comprehensive one, to blend two DVRIs, and to delete features in DVRIs without knowledge of the TF. As a semiautomatic method, our approach integrates user knowledge into the framework. Since the visualization goal highly depends on the tasks and users, our system can gain valuable input from users through the features they select and the editing operations they apply. Based on the DVRI editing operations, some advanced applications such as focus+context visualization with animation can be developed.

In our previous papers [51], [52], we proposed how to fuse features in different DVRIs and how to achieve focus+context visualization with animation. In this paper, we present a framework that is more general and can include more editing operations such as deleting and blending. To facilitate users to select features to edit, we integrate two semiautomatic feature selection tools, lazy snapping [24], and intelligent scissors [31] into our new system. We also introduce a method to edit DVRIs generated from different viewpoints. In addition, we describe how our method can facilitate TF design for expert users. More experimental results on real data have been conducted to demonstrate the effectiveness of our method.

This paper is organized as follows: After reviewing previous work in Section 2, we give an overview of our system in Section 3. DVRI editing operations and their corresponding energy functions are described in Section 4. We then present the image similarity metric in Section 5 and the TF producer in Section 6. After that, we demonstrate how the fusing operation can be used to facilitate TF design for expert users in Section 7.1 and how the blending operation can be exploited to generate animations for focus + context visualization in Section 7.2. Experimental results and discussions are presented in Section 8. We present our conclusions and suggest some future work in Section 9.

2 Related Work

2.1 Feature-Based Visualization

Weiskopf et al. [48] proposed several interactive clipping methods by exploiting the powerful capabilities of *Graphics Processing Unit* (GPU). Viola et al. [42] developed a novel importance-based approach for focus + context volume visualization, which was recently extended to allow automatic focusing on features in a volume [41]. Volume illustration techniques [3], [7], [22], [34], first introduced by

Rheingans and Ebert [35], provide an alternative way for focus + context visualization with abstraction techniques (for example, nonphotorealistic rendering). Bruckner and Gröller [3] and Correa et al. [7] proposed new volume manipulation techniques for illustration and visualization. Krüger et al. [22] presented a context preserving hotspot visualization technique to help users interactively visualize a volume data set. Wang et al. [45] developed an interactive GPU-assisted volume lens to magnify the regions of interest (ROI) while preserving the context by compressing other volume regions. Kim and Varshney [15] used a visual saliency-based operator to draw users' attention to the focus regions in a volume.

2.2 Transfer Function (TF) Design

An excellent survey on TF design can be found in [16]. There are two main categories of TF design methods, that is, *image-centric methods* and *data-centric methods* [32]. Marks et al. [29] developed an image-centric system that automatically generates many perceptually different images and organizes them efficiently for users to select. Kindlmann and Durkin [17] presented a histogram data structure to semiautomatically obtain good TFs with user guidance. The images generated by both approaches can be used as inputs to our system. Ma [28] proposed a novel approach based on image graphs to facilitate visual data exploration. König and Gröller [21] developed a TF design interface paradigm that provides several specification tools for each search domain. Although previous methods [21], [28] allow users to linearly combine TFs, they cannot always get the expected results (see Fig. 1). Kniss et al. [20] extended Kindlmann and Durkin's work by introducing different TF widgets and a novel dual-domain operation. More recently, Šereda et al. [44] presented a novel TF design method based on LH histograms, which can more easily distinguish features than the traditional histograms. Tory et al. [39] employed a parallel-coordinate-style interface to facilitate the exploration of TFs. Taking human factors into consideration, their method employs a history bar to help users backtrack to previous TF settings. Lum et al. [26] proposed an interactive multiscale TF design technique using filter banks that is particularly effective for classifying fuzzy or noisy features. Our method is based on the similarity of the rendered images and thus is an image-centric method.

2.3 Genetic Algorithms

Genetic algorithms are widely used in many fields like computer graphics [37]. He et al. [11] first employed GAs to generate TFs. Our approach is inspired by their work, but aims at editing DVRIs rather than designing TFs from scratch. 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 are to be retained or enhanced in the DVRIs. House et al. [12] used a GA to choose visualization parameters to optimize visualization quality.

2.4 Animated Visualization

Gagvani et al. [10] proposed a skeleton-based approach for volume animation. Their work was further extended by Chen et al. [5] with a novel spatial TF to specify deformation in volume animations. Lum et al. [27] presented an impressive visualization technique called *Kinetic Visualization* for creating motion along a surface to facilitate the understanding of static transparent 3D shapes. Weiskopf [47] described some relevant psychophysical and physiological findings to demonstrate the important role of color in the perception of motion. Correa and Silver [6] proposed a new data traversal approach for volumetric data with motion-controlled TFs. Three perceptual properties of motion: flicker, direction, and velocity have been thoroughly examined by Huber and Healey [13] in an experimental study. Although there has been growing interest in animated visualization, most previous works either aim at enhancing the perception of 3D shapes and structures of static objects [6], [27] or focus on evaluating some general motion attributes [13], [47]. Using animation for focus + context visualization is still an unexplored area.

3 System Overview

Fig. 2 shows the overview of our system. With this system, users are able to intuitively manipulate the features in DVRIs rather than in the complex TF domain. It allows users to fuse multiple features in distinct DVRIs into a comprehensive one, to blend two DVRIs, and/or to delete features from a DVRI.

Fig. 2a is the user interface of our system including the source DVRIs and their TFs, the target DVRIs and their TFs, and a history region. The source TFs represented as their corresponding DVRIs in the history region can be generated by any semiautomatic or automatic TF design approaches [17], [29]. Users can import the source DVRIs and their TFs from the history region at the bottom of the user interface by a simple drag-and-drop operation. For nonexpert users who are only interested in editing DVRIs, the TFs can be totally hidden from them because they do not need to directly operate on the TFs. The interface allows users to identify features in the source DVRIs with rectangles or other semiautomatic feature selection tools (see the red rectangles in the source DVRIs in Fig. 2a). Users can give each DVRI a different score ranging from 0 to 1. These scores can be viewed as the expected similarity values between the target DVRI and the source DVRIs. After that, our system begins to search for the target DVRI taking the expected similarity values into account.

Fig. 2b is the system architecture that consists of *energy function generator*, *TF producer*, *direct volume renderer*, and *image similarity evaluator*. Energy functions are used in stochastic search algorithms like GAs to evaluate the fitness of the candidate/intermediate solutions, which are generated in the optimization process. The energy function generator shown in Fig. 2b is a critical part of our framework. It aims at automatically creating an appropriate energy function based on users' editing operations on DVRIs. The generated energy function is then passed to the TF producer. The TF producer generates a number of intermediate TFs (candidate/intermediate solutions) using stochastic search algorithms and sends the intermediate TFs to the direct volume renderer for rendering corresponding intermediate DVRIs, which are then sent to the image similarity evaluator. The image similarity evaluator computes the image similarity value between each intermediate

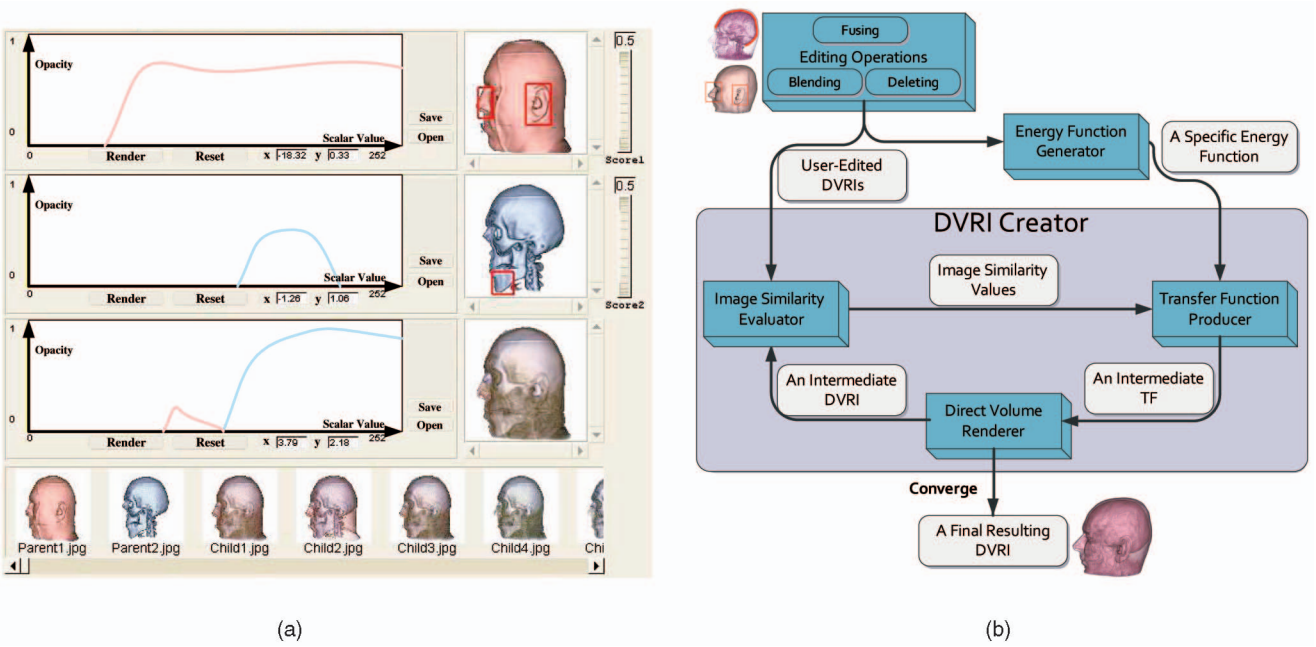

Fig. 2. The overview of our framework. (a) User interface that consists of source DVRIs and their TFs, target DVRI and its TF, and a history region. (b) System architecture that consists of energy function generator, TF producer, direct volume renderer, and image similarity evaluator.

DVRI and the source DVRIs and passes the value back to the TF producer. With the image similarity value, the energy function in the TF producer can compute the energy value of the intermediate TF. The smaller the energy value, the fitter the intermediate TF. Based on these energy values, the TF producer eliminates the unsuitable TFs and generates new and fitter TFs to begin the next cycle of refinement. When the stochastic search algorithm used in the TF producer converges, the system outputs the final DVRI rendered with the best found TF.

4 Editing Operations and Energy Functions

Our system supports the following editing operations: fusing multiple features in distinct DVRIs into a comprehensive one, blending two DVRIs into a DVRI with correct depth cues, and deleting certain features in a DVRI. These editing operations can be used together. For example, users can remove a feature from a DVRI and meanwhile fuse the DVRI with others.

As our system generates DVRIs using stochastic search algorithms, an energy function for evaluating the fitness of candidate solutions should be formed after users specify editing operations in DVRIs. In our system, we exploit an energy function based on image similarity and editing operations to objectively evaluate the fitness of intermediate TFs. Using the energy function, we are able to transform the DVRI editing problem to an optimization problem, that is, minimizing the energy function.

In this section, after discussing feature selection, we first introduce these editing operations and their corresponding energy functions. We assume that the source DVRIs are all rendered from the same viewpoint to simplify the presentation. Then, we present how to combine these editing operations together and how to extend our method so that these editing operations can be applied to DVRIs generated from different viewpoints. To simplify the presentation, we introduce the following terms: *Source image* for the source DVRI to be edited; *Target image* for the resulting DVRI after editing; *Source TF* for the TF of the source image; and *Target TF* for the TF of the target image.

4.1 Feature Selection

Our approach allows users to select desirable features in DVRIs to edit using rectangles or semiautomatic feature selection tools. Some features can be easily selected by directly drawing a rectangle around them. However, when the feature to be chosen is very close to other features, it cannot be simply selected using a rectangle. To solve the problem, we need accurate and quick feature selection tools. Fortunately, feature selection is one of the most widely studied problems in image processing and many excellent methods are available. To ease the task of selecting features with complex shapes, we use two semiautomatic feature selection tools, lazy snapping [24], and intelligent scissors [31] for our system. Lazy snapping can automatically snap the contour to the boundary of the feature with several paintings to indicate the inside and outside of the feature, whereas the intelligent scissors are able to trace the contour of the wanted feature using simple gesture motions with mouse. In our experiments, we found that lazy snapping can select a desirable feature faster than intelligent scissors, but intelligent scissors are more flexible and more capable of selecting only a part of the feature instead of the entire one. With the feature selection tools, our system can allow users to select a region containing the feature and this region will then be converted into a mask (see Fig. 5b) within which our image similarity evaluator performs. As our system employs a contour-based image similarity metric to evaluate the fitness of candidate TFs, and the contours of features are finally determined by *Canny edge detector*, users are not required to trace the

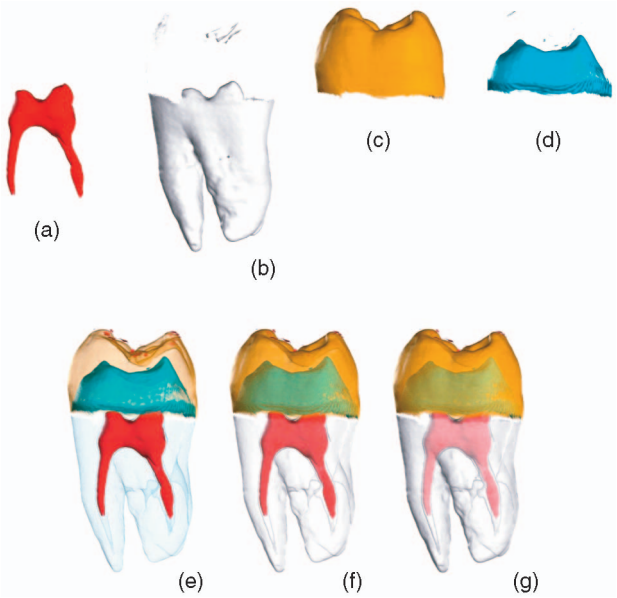

Fig. 3. The fusing operation: (a), (b), (c), and (d) Source DVRIs. (e) Target DVRI generated with $V_1 = 0.7$, $V_2 = 0.3$, $V_3 = 0.4$, and $V_4 = 0.6$. (f) Target DVRI generated with $V_1 = 0.5$, $V_2 = 0.5$, $V_3 = 0.5$, and $V_4 = 0.6$. (g) Target DVRI generated with $V_1 = 0.3$, $V_2 = 0.7$, $V_3 = 0.6$, and $V_4 = 0.4$.

contours of the feature exactly as long as the contours are inside the mask region, and the mask region does not contain a significant amount of contours from other features.

4.2 Fusing Operation

The fusing operation is used to combine multiple user-selected features that appear in different DVRIs into a comprehensive one. To apply this operation, users first identify and select multiple features from distinct DVRIs and vote for each feature or a whole DVRI to indicate their preference. The system then automatically generates a comprehensive DVRI containing all these selected features according to the votes. Each vote represents how much they would like a selected feature to be retained in the resulting DVRI. The energy function for the fusing operation is given as follows:

$$F_1 = \sum_{k=1}^{n} V_k * |V_k - S_k|, \tag{1}$$

where n is the number of source images, V_k represents the vote (or the score) given by users for the feature in source image k, and S_k denotes the computed image similarity value between the target image and the source image k. Both V_k and S_k ranges from 0 to 1. A more detailed definition of S_k can be found in (5) in Section 5. V_k, from another point of view, can be also considered as the similarity value expected by users between the target image and source image k. V_k also 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 target image gets proportional contributions from all the source images and is not dominated by any single one. Fig. 3 shows an example of fusing multiple features in distinct DVRIs (see Figs. 3a, 3b, 3c, and 3d) into three target DVRIs (see Figs. 3e, 3f, and 3g) with different user votes.

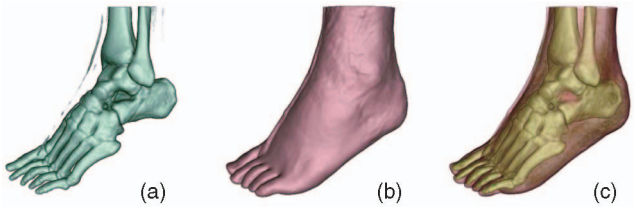

Fig. 4. The blending operation. (a) and (b) Source images with bone and skin, respectively. (c) Target image generated by blending (a) and (b) using our system.

4.3 Blending Operation

The blending operation is used to composite two DVRIs and generate a similar resulting image from alpha blending. In contrast to the traditional image blending operation, our DVRI blending operation can ensure the generation of true DVRIs. Compared with other artificially blended images, DVRIs usually contain correct depth cues and have less misleading information because DVR composites samples in depth sorted order. The energy function for the blending operation is defined as

$$F_2 = \alpha_1 * |\alpha_1 - S_1| + \alpha_2 * |\alpha_2 - S_2|, \tag{2}$$

where S_1 and S_2 are the computed image similarity values similar to S_k in (1), and α_1 and α_2 represent the alpha values used for blending. The blending operation can also be viewed as a special case of the fusing operation, that is, to fuse two whole DVRIs without user-selected features into a comprehensive DVRI according to the alpha values α_1 and α_2. Fig. 4c is a DVRI generated by blending Figs. 4a and 4b with $\alpha_1 = \alpha_2 = 0.5$ using our system.

4.4 Deleting Operation

The deleting operation allows users to remove extra features from a DVRI by selecting what they want to retain or to remove using lazy snapping (see Fig. 5a), a rectangle, or intelligent scissors. Suppose users want to remove several features in region A of a DVRI, and the rest of the DVRI is denoted as region B. The energy function for the deleting operation is given as follows:

$$F_3 = S_1 + (1 - S_2), \tag{3}$$

where S_1 is the computed image similarity value between the target image and the source image within region A, and

Fig. 5. The deleting operation. (a) A DVRI where the skin is to be removed. (b) Mask created from (a) using lazy snapping. (c) Resulting DVRI after executing the deleting operation.

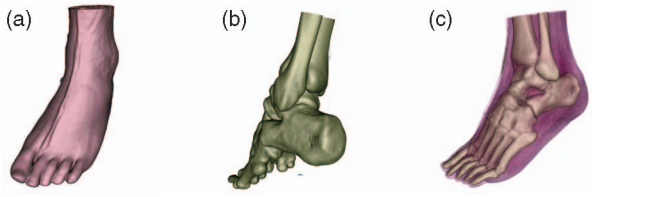

Fig. 6. The editing operation on DVRIs generated from different viewpoints: (a) and (b) Source images generated from different viewpoints. (c) Target image generated by blending the user selected features shown in (a) and (b).

S_2 is defined the same as S_1, but within region B. To minimize the energy function in (3), S_1 should be minimum and S_2 should be maximum, that is, the target image should look similar to the source image within region B, but differ from the source image within region A. Fig. 5 shows an example of the deleting operation. Lazy snapping is used to select features. Users select the feature to be retained by drawing several strokes (the yellow paintings in Fig. 5a) indicating the inside region of the feature and several strokes (the blue paintings in Fig. 5a) indicating the outside region. The other features will be removed. Fig. 5b is the mask obtained from lazy snapping. Fig. 5c is the resulting DVRI generated by our system.

4.5 Mix Multiple Editing Operations

Our system enables users to mix different basic operations together. For example, users can fuse multiple features in distinct DVRIs together and can meanwhile remove certain features from some DVRIs. To achieve this goal, we propose to use a comprehensive energy function F as follows:

$$F = F_1 + F_2 + F_3, \tag{4}$$

where F_1, F_2, and F_3 are the energy functions defined in (1), (2), and (3) for the fusing, blending, and deleting operations, respectively.

4.6 Editing Features from Different Viewpoints

Viewpoint selection is an important issue in scientific visualization [2], [38]. Appropriate viewpoints can reveal more 3D structures and enhance the comprehensibility of the data, whereas bad viewpoints may deliver less informative results and may even present misleading information. Thus, it is advantageous and important for the system to allow users to edit features in DVRIs rendered from different viewpoints and then render the result from a good viewpoint so that more informative results can be revealed at the same time. Our framework can be easily extended to handle this situation.

Given n viewpoints and their corresponding DVRIs, our system allows users to select a common good viewpoint either manually or by some automatic viewpoint selection approaches [2], [4], [38]. We then rerender the user selected features from this viewpoint. For any unwanted features, our system automatically removes them using the deleting operation. After that, the basic DVRI editing operations can be directly applied to these new DVRIs because they are now generated from the same viewpoint. Figs. 6a and 6b are two DVRIs rendered from different viewpoints and each of them contains one salient feature, that is, the skin of the toes and the heel bone, respectively. Fig. 6c is the result of blending Figs. 6a and 6b from a suitable viewpoint where all the salient features are clearly revealed.

5 Image Similarity

Image similarity is an important but difficult problem in various fields of computer science. Many approaches have been established to compute the similarity of two images. They can be generally classified into three categories, that is, spatial domain metrics, spatial-frequency domain metrics, and perceptually-based metrics [53]. However, image similarity evaluation is still an open problem and far from being completely solved.

Most previous approaches have some nice properties and advantages, but they also suffer from some drawbacks and cannot work well in all situations. For instance, image similarity methods using mutual information like [43] do not consider neighborhood regions of corresponding pixels and therefore ignore the spatial information inherent in images. Peak Sinal-to-Noise Ratio (PSNR) and Mean Squared Error (MSE) are two other widely used spatial domain metrics. However, they fail to correlate well with perceived similarity measurement [46] and are unable to detect movement between pixels [33]. Recently, there has been increasing interest in perceptually-based image comparison metrics such as Visual Differences Predictor (VDP) [8], [49] and Visual Discrimination Model (VDM) [25]. Although they perform generally better than other methods, they are still not robust and suffer from difficulties in finding a correct general Contrast Sensitivity Function (CSF) representation that is used to calculate and weigh the errors between the two signals to be compared [23]. It is observed that none of the sophisticated methods can clearly surpass those simple spatial domain metrics like PSNR under strict testing conditions [46]. Moveover, they gain advantage over those simple methods at the cost of time performance or memory.

In order to strike a balance between performance and cost, we develop a contour-based similarity metric to compare two DVRIs. The contour is one of the most important perception clues for 3D structures. It is able to simplify the representation of an object while retaining significant features (that is, the shape of an object). Thus, it can be used to compare the similarity of objects. Our system first converts the DVRIs to gray-scale images and detects the edge images from the gray-scale images using the well-established optimal edge detector —*Canny edge detector*. Since most contours appearing in the DVRIs also appear in gray-scale images, our contour-based metric still works well. To compare two edge images, a straightforward method is to directly compute the differences pixel by pixel. However, contours detected from DVRIs using a specific TF may shift several pixels when the TF is slightly changed. If we simply compare the images pixel by pixel, then we cannot detect a pair of contours that look very similar, but whose actual positions do not match perfectly (for example, shifting one or two pixels). To handle this problem, we first apply the Gaussian filter 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 DVRIs and 3×3 for 256×256 DVRIs. Then, we can quickly compute the image similarity pixel by pixel.

The image similarity value S_k between the target edge image and the source edge image k is computed as follows:

$$S_k = \frac{\sum_{y=1}^{height}\sum_{x=1}^{width} Q(x,y)}{N_{source}}, \quad (5)$$

$$Q(x,y) = \begin{cases} 1 & \text{if } P(x,y) < threshold \\ 0 & Otherwise, \end{cases} \quad (6)$$

$$P(x,y) = |K_{(x,y)} - K'_{(x,y)}|, \quad (7)$$

where N_{source} in (5) is the number of all pixels on the edges of the source k's edge image. $threshold$ in (6) is a parameter set by the system. $P(x,y)$ in (7) represents the difference between two corresponding pixels. K and K' are the Gaussian filtered target and source edge images with resolution $(width, height)$.

Notice that we consider only the pixels on the source edge image k for S_k, that is, (5) is only computed if $K'_{(x,y)} \neq 0$. Thus, $\sum_{y=1}^{height}\sum_{x=1}^{width} Q(x,y) \leq N_{source}$ and S_k in (5) ranges from 0 to 1. If image similarity is only evaluated within a mask region of the whole image, our system considers only the pixels within this region. Thus, the N_{source} in (5) only counts the number of edge pixels within this region.

To determine the default $threshold$ in (6), we conducted extensive experiments on real-volume data. All the edge images generated are 8-bit gray-scale images. From the experiments, we found that when the $threshold$ ranges from 60 to 90, desirable results can be achieved. Based on this, the default value of $threshold$ is set to be 80 in our system, which can be adjusted by users.

6 Transfer Function Producer

The TF producer is the most important component in our system. It is used to generate TFs for desired DVRIs based on the energy function created by the energy function generator.

6.1 Genetic Algorithm

A *Genetic Algorithm* (GA) is a search algorithm imitating the process of natural evolution [30]. It is particularly useful for searching solutions to optimization problems, especially when the search space is huge and unknown. It can also quickly find a sufficiently good solution, which may not be the global optimum. To use GAs, candidate solutions to a specific problem must be represented as *genomes* such as binary encodings, tree encodings, or floating number encodings. Similar to natural selection and natural genetics, the algorithm initially creates a possible set of genomes called a *population or generation*. The population evolves with the *crossover* (or *reproduction*) operator and the *mutation* operator over multiple generations and generates better and better genomes. The pseudocode for the GA used in our system is shown in **Algorithm 1**.

Algorithm 1. Simple Genetic Algorithm

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 the roulette wheel scheme as source 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

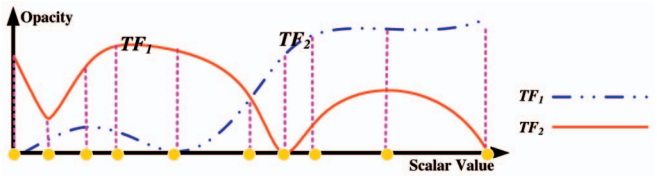

Fig. 7. The genome representation for 1D TFs.

6.2 Initial Population

The initial population of the algorithm can be treated as an initial guess of the desirable results. As the following evolution is based on the initial guess (that is, the following populations evolve from the initial population), the initial population may greatly affect the performance of the algorithm. An appropriate one may make the algorithm converge early and obtain the desirable results quickly. According to our experiments, the GAs that begin with the initial population created by the linear combination of the source TFs with random weights can generally find the desirable target TFs faster than those with a completely random population. Our system therefore creates the initial population by linearly combining the source TFs with random weights.

6.3 Genome Representation

The solution encoder/decoder specifies the genome representation by analyzing the source TFs. Aimed at defining the optimal genome representations [30], our approach represents a TF as a 1D array of floating-point numbers. For the 1D source TFs, the component first smoothes the source 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 Fig. 7, there are two source TFs denoted by different line patterns. The points on the axis of the scalar value are the union of the sampling positions of the source TFs. The vertical dashed lines start with 0 and end with the maximum opacity value of the source TFs. They act as the range of opacity values of the corresponding points on the axis of the scalar value. All the opacity values at the aforementioned points (on the axis of the scalar value) of each candidate target TF constitute a genome.

Our system can also encode/decode 2D source TFs. Fig. 8 shows the 2D source TFs constructed with the scalar value and gradient magnitude of the volume. They are based on Kniss et al.'s triangular classification widget

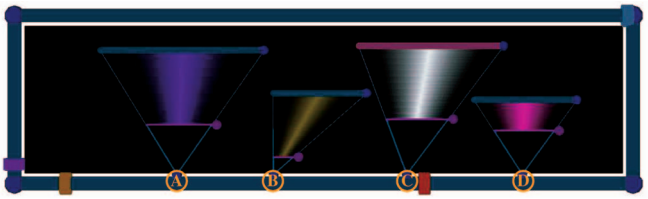

Fig. 8. The genome representation for 2D TFs.

(details can be found in [20]). Each inverted triangle widget is used to distinguish a distinct feature. We assume that the position and the shape of each widget does not change. As opacity is maximal along the centerline of the widget and linearly ramps down to both sides, the opacity distribution within each widget can be represented by the maximum opacity in this widget as a normalized real value ranging from 0 to 1 [20]. Thus, we can denote the TFs with an array of real numbers such as A, B, C, and D shown in Fig. 8, indicating the opacity distribution of each widget. Each element of the array is bounded from 0 to 1. The genome representation for this case is thus the union of all such arrays without any overlap.

6.4 Selection Scheme

The selection scheme determines how to choose genomes in the current population to produce offsprings. The selection scheme in GAs imitates natural selection in such a way that better (or fitter) genomes are more likely to survive and be selected to produce their offsprings. The selection has to strike a balance between a too-weak and a too-strong selection (or between "exploitation" and "exploration") [30]. A too-strong selection makes the suboptimal genomes dominate the population and the evolution converge too early, whereas a too-weak selection leads to a too-slow convergence. Many different selection methods have been developed to avoid too-strong selections and too-weak selections in the GA community. Our system employs one of the most commonly used selection schemes called *roulette wheel*.

6.5 Evolution Operators (Crossover and Mutation)

Crossover combines two source genomes with probability p_c to produce two new offsprings in the hope of preserving features of the parents. In our system, we use the *two-point crossover* method, which selects two random positions on each source and exchanges the segments between them. After crossover operations, the GAs perform mutation for each genome with low probability p_m similar to natural mutation. The purpose of the mutation is to increase the diversity of the population without converging too early (or premature convergence). For example, GAs usually tend to converge to a local optimum while evolving. With the help of mutation, GAs are more likely to get out of the local optimum.

6.6 Parameters for GAs

It is important to appropriately set parameters such as population size, crossover rate, and mutation rate for the GAs so that the algorithms can perform well in practice. However, these parameters are difficult to optimize due to their nonlinear dependency. There are no general guidelines regarding the setting of these parameters in different situations. Early work by Jong [14] gives some suggestions about the best parameters: the best population size is around 50 genomes; the best crossover rate is about 0.6; and the best mutation rate is 0.01. These parameters are widely used.

7 Applications

Our DVRI editing techniques can be used for various applications like interactive TF design and focus+context visualization.

7.1 Interactive Transfer Function Design

Our framework can be used not only as an image-editing tool for nonexpert users, but also as an interactive TF design method for users with expertise in TF design. As an image-editing tool, our system hides the generated TFs from end users who will only deal with DVRIs, whereas as a TF design method, our system exposes the TFs to users, and the users can directly manipulate them to gain insight into data. In this section, we discuss our framework from the viewpoint of TF design.

TF design is critically important and is a very challenging issue in DVR. It has been a hot research topic in recent years and many excellent approaches have been proposed. Most of these approaches focus on reducing the search space for ideal TFs and developing intuitive user interfaces. However, the resulting TFs from these methods may still not be good enough and further adjustments may be needed. It is not intuitive for users to directly manipulate TFs in the TF domain.

As it is relatively easier for users to generate good TFs for different features of volumetric data, we propose TF fusing, which merges several TFs into a comprehensive one, as a new technique for TF design. Our TF design method is based on fusing features in DVRIs. Suppose n TFs along with the corresponding n DVRIs have been generated by automatic or semiautomatic methods. Then, in each DVRI, users can select features they want and vote for these features to show their preferences. Then, the TF design problem is converted to fusing features in different DVRIs and the TF for the final DVRI is outputted as the comprehensive TF that users look for.

Our approach has some advantages. First, we simplify TF design by converting it into a TF fusing problem. Our approach can design TFs from simple to complex, which can be easily grasped by users. Our system is more intuitive for users because they can work directly on DVRIs instead of in the TF domain. Second, users are able to exploit previous spent efforts represented as saved TFs to obtain a better or comprehensive new TF. By allowing users to vote for TFs and select ROIs in the images rendered by the TFs, our system can gain valuable input from users and keep desirable features in previous good TFs while subjugating unimportant features in the data. Third, our method can be used together with other TF design methods and may complement their strengths. The output TFs from other popular TF design methods such as semiautomatic TF generation [17] or TF design galleries [29] can serve as inputs to our method. These TFs can be fused with other TFs such as manually tuned TFs by users to generate a comprehensive TF for data exploration or presentation.

7.2 Animation for Focus + Context Visualization

In this section, we exploit the DVRI editing techniques to generate animations for focus + context volume visualization. Our work is motivated by an interesting psychological phenomenon—"*bird in cage*"—also called *afterimage* [36]. It is closely related to the theory of *persistence of vision*, which accounts for the illusion of motion in the film industry. According to the theory, a visual image will persist in a user's brain for a short time after the real image has disappeared from his/her eyes. Therefore, if we quickly show a sequence of keyframes and intermediate frames containing the focus and context, then the focus and context can simultaneously appear in users' brains and users can figure out their 3D relationship. The spatial relationship between the focus and the context is then revealed in the animation process. To generate animations for DVRIs with large transparent areas, depth cues and smooth transitions are especially important for conveying correct information. As mentioned before, if we directly apply traditional animation techniques such as the alpha blending technique to DVRIs, we may not get the correct depth cues and some misleading information may be introduced [1]. To address these problems, we propose a novel data-centric animation technique using our DVRI editing framework for the intermediate frame generation. As all frames including keyframes and intermediate frames of animations are generated by DVR, correct depth cues are provided and misleading information is minimized. In addition, our method applies the blending operation provided by our framework to generate a sequence of intermediate frames (DVRIs) from any two successive keyframes (DVRIs), which can guarantee the smooth transitions between frames as the operation itself is based on image similarity. We further develop two animation editing techniques, *level of detail (LOD)* and *zoom in/out*, to emphasize the focus frames and suppress the context frames in the time domain of the animation.

7.2.1 Similarity-Based Tweening between Keyframes

We assume that multiple keyframes and their TFs have already been generated by users. These keyframes either emphasize the focus or the context of the data. Suppose that the TFs for two successive keyframes, P_1 and P_2, are TF_{P_1} and TF_{P_2}, and the number of intermediate frames between two keyframes is N. To tween between these two keyframes, one straightforward solution is to linearly interpolate the key TFs to obtain a number of intermediate TFs, which can then be used to generate a sequence of intermediate frames by DVR. For example, the TF for the ith intermediate frame can be computed as $TF_i = \frac{(N-i)}{N} * TF_{P_1} + \frac{i}{N} * TF_{P_2}$. However, as we mentioned before, this approach may not get the expected results. Based on our experiments with real data (see Figs. 15a, 15b, 15c, 15d, and 15e), this method cannot generate smooth transitions between frames. As the compositing operation used in volume rendering is nonlinear, the smooth transitions between two TFs cannot guarantee the smooth transitions between the resulting DVRIs.

To solve this problem, we propose a similarity-based tweening technique using our blending operation. To guarantee a smooth animation, for the ith intermediate frame P_i, the similarity value between P_i and P_1 should be $\frac{(N-i)}{N}$ and the similarity value between P_i and P_2 should be $\frac{i}{N}$. In our data-centric method, we compute a series of intermediate keyframes using the blending operation with linearly changed α values.

7.2.2 Animation Editing

Each keyframe used in the animation usually contains only one specific feature except in a few situations in which it is difficult to separate that feature from other features. If there are many features in the volumetric data, there will be a number of keyframes in the animation. Thus, the total number of frames including the keyframes and intermediate frames is too large, which makes it difficult for users to focus on the frames containing the features they want to see. To overcome this problem, we propose two techniques, LOD and zoom in/out, to emphasize the important frames while suppressing the context frames. The proposed techniques are applied on the *timeline* of the animation, which controls the display time of each frame.

In our paper, we apply LOD to the timeline of the animation to make the exploration easier and more flexible in a visualization process. Users are allowed to select multiple successive keyframes and combine them into a new keyframe using the fusing operation with different user votes. After that, the system replaces these keyframes with the newly combined keyframe for the animation. Tweening is automatically generated for these new keyframes. Users can perform this combination many times until they obtain a satisfactory animation. In our work, we mainly use the LOD technique to reduce the number of the context keyframes.

After the context keyframes are clustered by the LOD technique, users may need to further emphasize the frames in which they are interested. In order to highlight these focus frames, we apply a technique similar to the fisheye-view approach [9] to the timeline of the animation, which gives the important frames a longer display time and reduces the display time for less important frames.

8 Experiment Results and Discussions

Our system was implemented in C++ based on the Visualization Toolkit (VTK) 5.0 library [18] and the code provided by Kniss et al. [19]. We tested the system on a Pentium(R) 4 3.2-GHz PC with 2 Gbytes of RAM and an Nvidia Geforce 8800 GTX GPU with 768 Mbytes of 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 optimization process and was then switched to 512×512 for rendering the final results. All features in this section were selected by users using rectangles, lazy snapping, or intelligent scissors. For features selected by lazy snapping (see Figs. 11 and 14), yellow strokes are uniformly used to indicate the inside regions of the features, whereas blue strokes the outside regions. In the following experiments, we obtained sufficiently good results within acceptable time frames (around 4 to 15 seconds).

8.1 Experiments for the Basic Editing Operations

To demonstrate the effectiveness of the basic editing operations, we carried out several experiments on different

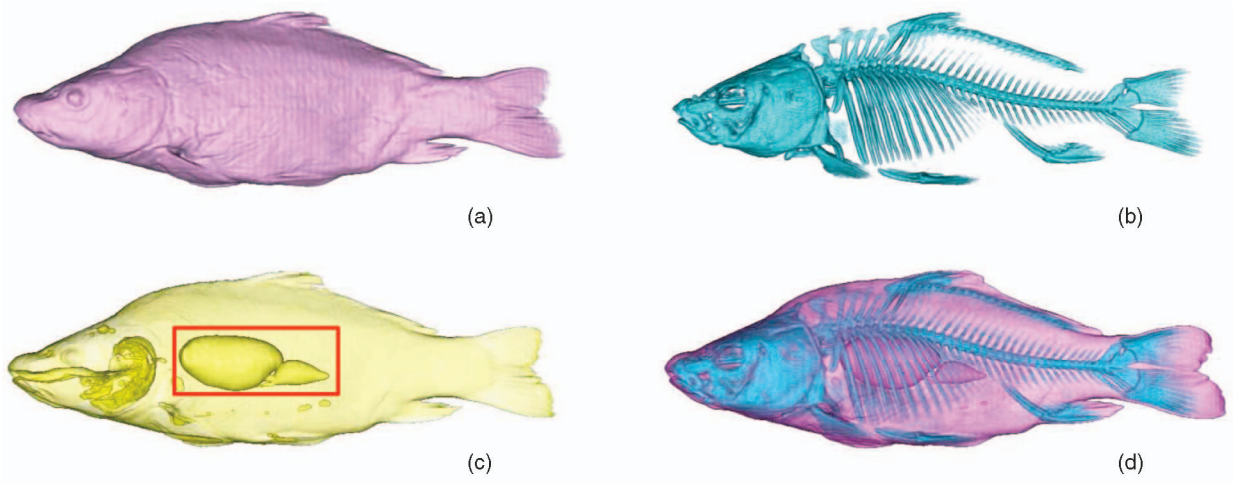

Fig. 9. Experiment on a CT carp data set. (a) Source image 1. (b) Source image 2. (c) Source image 3 with a user-selected feature indicated by a rectangle. (d) Target image obtained by fusing features in (a) with 0.3 scores, features in (b) with 0.4 scores and the user-selected feature in (c) with 0.3 scores.

volumetric data. Figs. 3 and 9 show two examples of the fusing operation. Users are allowed to specify how much they want the features in source DVRIs to appear in the resulting DVRIs with different user votes. Thus, our approach, from another point of view, can also be treated as an approach for importance-based visualization. Users just simply select some important features and provide an importance value (the user vote) for each feature or the whole DVRI. Our method then automatically generates an image revealing these regions. The DVRIs in Fig. 3 were generated from a computerized tomography (CT) human tooth volume data ($(256 \times 256 \times 161)$, whereas the DVRIs in Fig. 9 were rendered from a CT carp data set $(256 \times 256 \times 512)$.

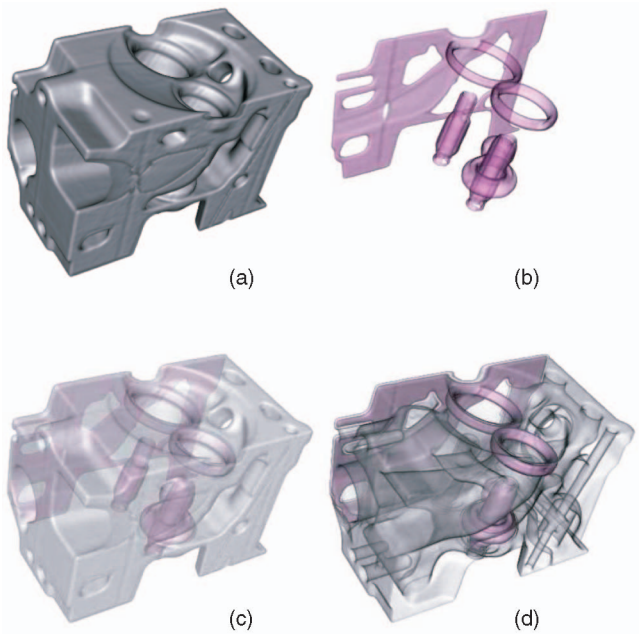

Fig. 10. Experiment on a CT engine data set. (a) and (b) Source images to be blended. (c) Target image generated by the traditional blending operation. (d) Target image generated using our proposed blending operation.

We carried out the second experiment on a CT engine data set $(256 \times 256 \times 128)$ to validate the effectiveness of the blending operation. Figs. 10a and 10b are two DVRIs to be blended with $\alpha_1 = \alpha_2 = 0.5$. Fig. 10c was created by the traditional blending operation, whereas Fig. 10d was generated in 7 seconds using our method. From the figures, we can see that the result in Fig. 10d provides correct depth cues. In contrast, some depth cues and the details inside the engine are missed in Fig. 10c.

Our third experiment was for the deleting operation on DVRIs. Fig. 11a shows one DVRI where features to be retained were selected by users using lazy snapping. The other features should be removed. Fig. 11b shows the resulting image after the deleting operation.

Our system was also tested on a large volume data with resolution $512 \times 512 \times 626$, as shown in Fig. 12. Fig. 12c was generated by blending Figs. 12a and 12b both with 0.5 scores in 13 seconds.

8.2 Experiments for Transfer Function Design

We demonstrate that TFs generated by other TF design methods can be fine tuned by our TF design method. In the first example, we fused a TF generated by the semiautomatic method [17] and a TF tuned by users to make the contours in the DVRI clearer. In the second example, we

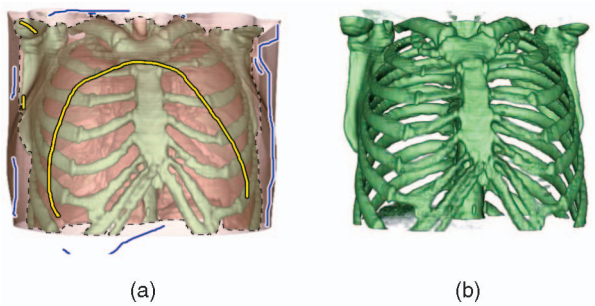

Fig. 11. Experiment on a CT breast data set. (a) A DVRI with multiple layers and features selected by lazy snapping. (b) Target image generated by our editing system without the unwanted tissue.

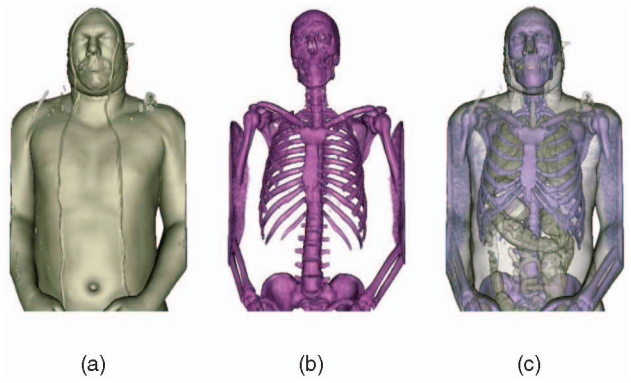

Fig. 12. Experiment on a CT human body data set. (a) Source image 1. (b) Source image 2. (c) Target image obtained by blending (a) and (b) with 0.5 scores, respectively.

removed some unwanted features in the DVRI generated from the semiautomatic TF design method and outputted the resulting TF for further processing.

8.2.1 Make the Contours Clearer

The experiment on an magnetic resonance imaging (MRI) head data set $(256 \times 256 \times 256)$ is to demonstrate that our approach is able to generate a fused image containing the user selected features with clearer contours. The image shown in Fig. 13b was created by the semiautomatic TF design method [17]. However, the semiautomatic method could not easily obtain an image revealing the inner structures of the head. Thus, we manually created an image (see Fig. 13a) with a simple TF capable of revealing the inner structures of the data. However, this image was not as good as 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 to be retained in Fig. 13a using intelligent scissors and features to be retained in Fig. 13b with rectangles. 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. Fig. 13c shows the result generated in 7 seconds.

8.2.2 Remove Unwanted Structures

Fig. 14a was generated by the semiautomatic TF design method. However, there is a superfluous structure surrounding the turtle, as the semiautomatic method views the structure also as a feature. To remove it, users have two

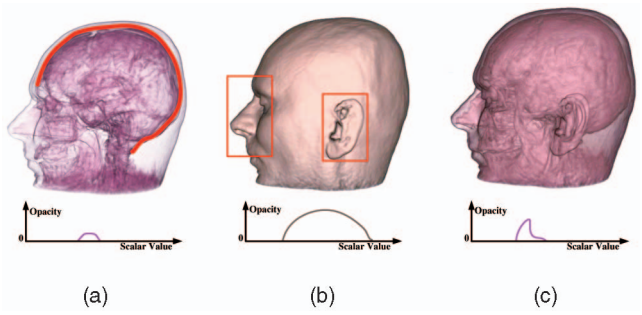

Fig. 13. Experiment on an MRI human head data set. (a) and (b) Source images with user selected features. (c) Target image generated by fusing the user selected features shown in (a) and (b).

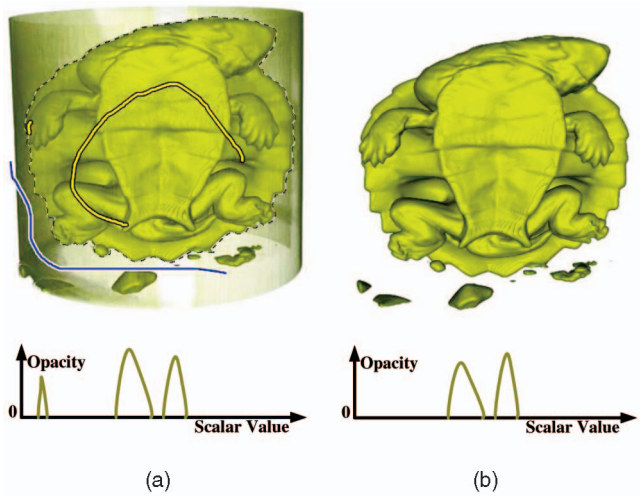

Fig. 14. Experiment on a CT Turtle data. (a) A DVRI generated by the semiautomatic TF design method with features selected by lazy snapping. (b) Target image created by erasing other unwanted structures.

choices. The first is to set the ambient gradient threshold [17] bigger than 0 and the other is to manually fine tune the generated TF in the TF domain. Obviously, setting the threshold is a cumbersome job since users without expertise may have to try many times to find a suitable threshold. On the other hand, erasing it by directly manipulating TFs in the TF domain may be an easy job for TF experts, but it is not intuitive and may even be difficult for nonexpert users, as they may not be familiar with the TF domain. In contrast, our method can supplement the strength of this excellent TF* design method, as it allows users to indicate which features they want to retain or remove by lazy snapping and our system can automatically remove the unwanted features, which is quite straightforward.

8.3 Experiments for Generating Animations

Fig. 15 shows the differences of the tweenings generated by different methods. Figs. 15p and 15q are the keyframes indicated as *keyframe1* and *keyframe2*, and Figs. 15a, 15b, 15c, 15d, and 15e were generated by linearly interpolating the TFs of the keyframes ($TF = \alpha * TF_1 + \beta * TF_2$, where α and β are shown below the corresponding figures, and TF_1 and TF_2 are shown in Fig. 15r). Figs. 15a, 15b, 15c, 15d, and 15e fail to form smooth transitions from *keyframe1* to *keyframe2*. The change from Fig. 15a to 15b is too abrupt, and Figs. 15b, 15c, 15d, and 15e are almost the same as *keyframe2* (Fig. 15p). In an opposite manner, Figs. 15f, 15g, 15h, 15i, and 15j generated by our data-centric method (that is, the proposed blending operation) and Figs. 15k, 15l, 15m, 15n, and 15o created by our image-centric method described in [51] have smoother transitions between the successive keyframes. Additionally, the tweening (Figs. 15f, 15g, 15h, 15i, and 15j) generated by our data-centric method is the best among all these methods. Fig. 15k and 15o make the morphing, generated by the image-centric method, a bit abrupt and not as good as that created by the data-centric method, since they make the transition from *keyframe1* to Fig. 15l and the transition from Fig. 15n to *keyframe2* rough. Moreover, the images created by the data-centric method have richer details and provide better depth cues than those created by

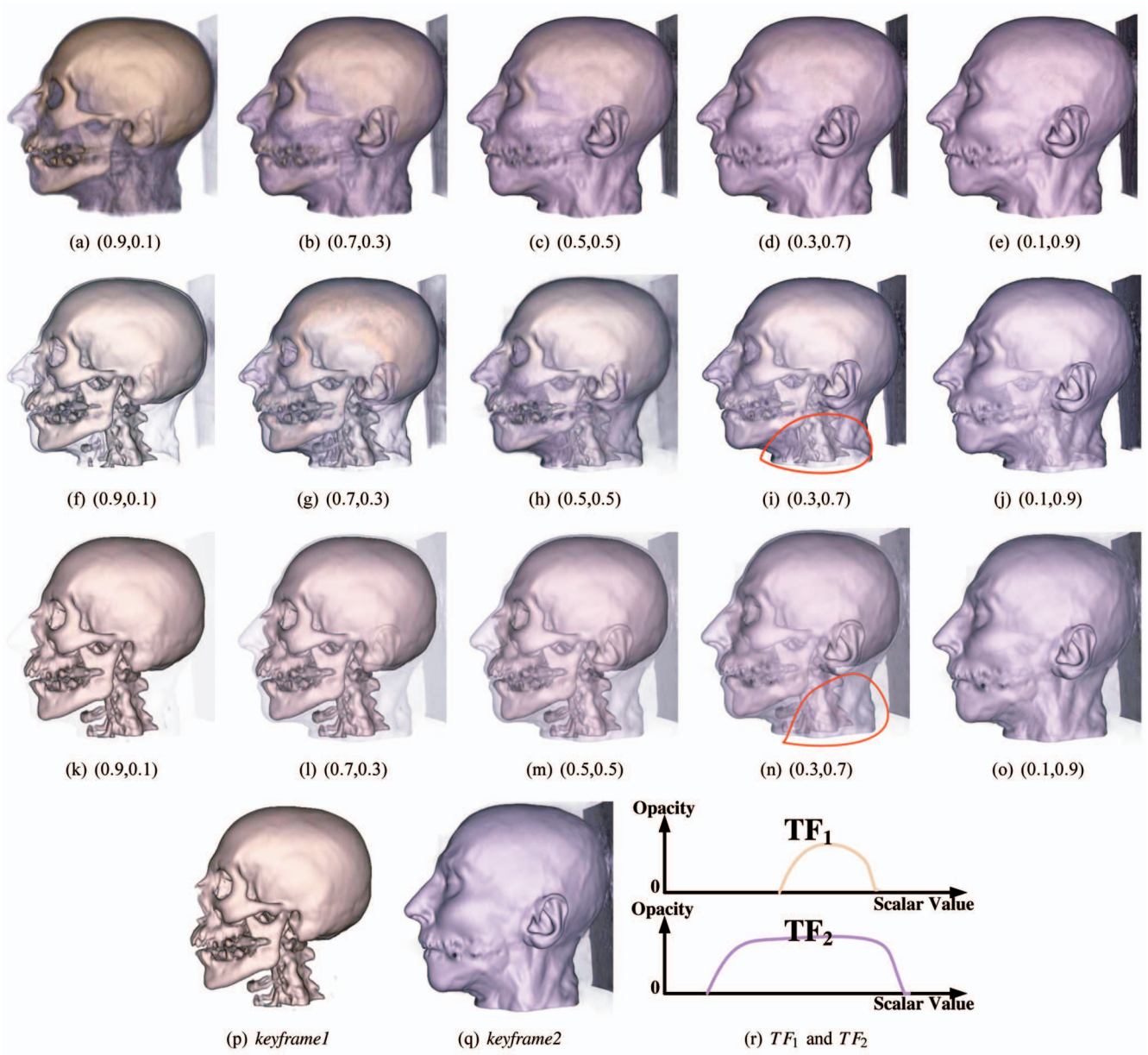

Fig. 15. Tweening between keyframes. (p) and (q) Keyframes. (a), (b), (c), (d), and (e) Intermediate frames created by linearly interpolating TFs of the keyframes with different (α, β). (f), (g), (h), (i), and (j) Intermediate frames created by the data-centric blending operation with different (α_1, α_2). (k), (l), (m), (n), and (o) Intermediate frames generated by the image-centric blending operation with different (α_1, α_2). (r) TFs of the keyframes.

the image-centric method (see the regions selected by the red curves in Fig. 15i and 15n).

8.4 Discussions

Our system works well for most volume data we tested. However, as it uses a contour-based metric to evaluate the similarity value, it may not be able to obtain desirable results for volume data, which contain complex features without well-defined contours. For example, we tested our approach on a CT scan of a bonsai tree $(256 \times 256 \times 256)$. For the blending operation, our approach failed to generate a good result like Fig. 16d and can only create some results like Fig. 16c, which does not present the features in Fig. 16b. This is because the leaves of the bonsai do not have clear contours. To obtain a desirable result and avoid such problem, we can choose the well-defined features having clear contours instead of the original whole images and fuse these features into a new image. With this trick, we selected only the trunk of the tree in Fig. 16a and fused it with Fig. 16b using our system and finally obtained the resulting image shown in Fig. 16d.

Our system is able to use other existing TF design methods as the front end. It can therefore take advantage of existing research and can complement their strengths by introducing some useful DVRI editing operations. However, our system also suffers from the drawbacks of those methods. For instance, if the method which our system is built on is not capable of revealing a specific feature, our system cannot reveal that feature either.

9 Conclusions and Future Work

In this paper, we presented a novel framework based on image similarity and GAs, which allows users to directly

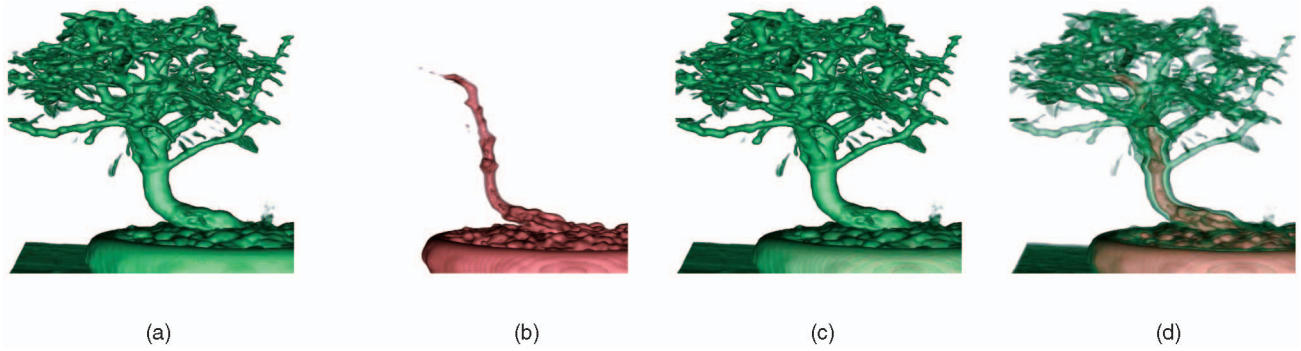

Fig. 16. Experiment on a CT bonsai tree. (a) Source image 1. (b) Source image 2. (c) Target image obtained by blending (a) and (b) with 0.5 scores, respectively. (d) Target image obtained by fusing (b) and the trunk in (a) together with 0.5 scores, respectively.

edit features in DVRIs and interactively design TFs. The system provides users with three basic editing operations: fusing multiple features in distinct DVRIs into a comprehensive one, blending two DVRIs into a DVRI with correct depth cues, and removing features from a DVRI. Based on these editing operations, smooth animations for focus + context volume visualization can be created. Our system provides an intuitive method for users without expertise in TF design to effectively explore volumetric data. The framework is flexible and easy to extend. For example, the GAs can be replaced by some other optimization algorithms and the image similarity evaluator can use some other image similarity metrics.

There are many possible venues for future work. Our implementation was not highly optimized for performance. The average runtime for the editing operations on the tested volumetric data sets was around 7 seconds. We plan to exploit GPU-accelerated GAs [50] and more advanced GPU-based DVR techniques to further reduce the runtime of our algorithm. Our energy function highly depends on the image similarity metric, which is a complicated issue involving human perception. The contour-based image similarity metric may not work well for volume data, which contain features without well-defined contours. We plan to develop more sophisticated image similarity metrics for complex transparent DVRIs and test them with more real volume data sets. We will also investigate other DVRI editing operations and integrate them into our framework.

ACKNOWLEDGMENTS

This work was supported by RGC Grant CERG 618705 and HKUST Grant DAG 04/05 EG02. The authors would like to thank the anonymous reviewers for their valuable comments and thank Hong Zhou and Ming-Yuen Chan for their help.

REFERENCES

[1] P. Baudisch and C. Gutwin, "Multiblending: Displaying Overlapping Windows Simultaneously without the Drawbacks of Alpha Blending," *Proc. SIGCHI Conf. Human Factors in Computing Systems,* pp. 367-374, 2004.

[2] U. Bordoloi and H.-W. Shen, "View Selection for Volume Rendering," *Proc. IEEE Visualization Conf.,* pp. 487-494, 2005.

[3] S. Bruckner and M.E. Gröller, "Exploded Views for Volume Data," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 5, pp. 1077-1084, Sept./Oct. 2006.

[4] M.-Y. Chan, H. Qu, Y. Wu, and H. Zhou, "Viewpoint Selection for Angiographic Volume," *Proc. Second Int'l Symp. Visual Computing (ISVC '06),* pp. 528-537, 2006.

[5] M. Chen, D. Silver, A.S. Winter, V. Singh, and N. Cornea, "Spatial Transfer Functions: A Unified Approach to Specifying Deformation in Volume Modeling and Animation," *Proc. Volume Graphics Conf.,* pp. 35-44, 2003.

[6] C. Correa and D. Silver, "Dataset Traversal with Motion-Controlled Transfer Functions," *Proc. IEEE Visualization,* pp. 359-366, 2005.

[7] C. Correa, D. Silver, and M. Chen, "Feature Aligned Volume Manipulation for Illustration and Visualization," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 5, pp. 1069-1076, Sept./Oct. 2006.

[8] S. Daly, "The Visible Difference Predictor: An Algorithm for the Assessment of Image Fidelity," *Digital Images and Human Vision,* A.B. Watson, ed., chapter 14, pp. 179-206, MIT Press, 1993.

[9] G.W. Furnas, "Generalized Fisheye Views," *Proc. SIGCHI Conf. Human Factors in Computing Systems,* pp. 16-23, 1986.

[10] N. Gagvani, D. Kenchammana-Hosekote, and D. Silver, "Volume Animation Using the Skeleton Tree," *Proc. IEEE Symp. Volume Visualization,* pp. 47-53, 1998.

[11] T. He, L. Hong, A. Kaufman, and H. Pfister, "Generation of Transfer Functions with Stochastic Search Techniques," *Proc. IEEE Visualization,* pp. 227-234, 1996.

[12] D. House, A. Bair, and C. Ware, "On the Optimization of Visualizations of Complex Phenomena," *Proc. IEEE Visualization,* pp. 87-94, 2005.

[13] D.E. Huber and C.G. Healey, "Visualizing Data with Motion," *Proc. IEEE Visualization,* pp. 527-534, 2005.

[14] K.A.D. Jong, "An Analysis of the Behavior of a Class of Genetic Adaptive Systems," PhD dissertation, Dept. of Computer and Comm. Science, Univ. of Michigan, 1975.

[15] Y. Kim and A. Varshney, "Saliency-Guided Enhancement for Volume Visualization," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 5, pp. 925-932, Sept./Oct. 2006.

[16] G. Kindlmann, "Transfer Functions in Direct Volume Rendering: Design, Interface, Interaction," *Proc. ACM SIGGRAPH Course Notes,* 2002.

[17] G. Kindlmann and J.W. Durkin, "Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering," *Proc. IEEE Symp. Volume Visualization,* pp. 79-86, 1998.

[18] Kitware, *The Visualization Toolkit,* www.vtk.org, 2006.

[19] J. Kniss, G. Kindlmann, and C. Hansen, *Simian,* http://www.cs.utah.edu/~jmk/simian/index.htm, 2002.

[20] J. Kniss, G. Kindlmann, and C. Hansen, "Multidimensional Transfer Functions for Interactive Volume Rendering," *IEEE Trans. Visualization and Computer Graphics,* vol. 8, no. 3, pp. 270-285, May/June 2002.

[21] A.H. König and E.M. Gröller, "Mastering Transfer Function Specification by Using VolumePro Technology," *Proc. Spring Conf. Computer Graphics,* pp. 279-286, 2001.

[22] J. Krüger, J. Schneider, and R. Westermann, "Clearview: An Interactive Context Preserving Hotspot Visualization Technique," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 5, pp. 941-948, Sept./Oct. 2006.

[23] B. Li, G.W. Meyer, and R.V. Klassen, "A Comparison of Two Image Quality Models," *Proc. SPIE Conf. Human Vision and Electronic Imaging III,* vol. 3299, pp. 98-109, 1998.
[24] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum, "Lazy Snapping," *ACM Trans. Graphics,* vol. 23, no. 3, pp. 303-308, 2004.
[25] J. Lubin, "A Visual Discrimination Model for Imaging System Design and Evaluation," *Vision Models for Target Detection and Recognition,* pp. 245-283, World Scientific Publishing, 1995.
[26] E.B. Lum, J. Shearer, and K.-L. Ma, "Interactive Multi-Scale Exploration for Volume Classification," *The Visual Computer,* vol. 22, nos. 9-11, pp. 622-630, 2006.
[27] E.B. Lum, A. Stompel, and K.-L. Ma, "Using Motion to Illustrate Static 3D Shape—Kinetic Visualization," *IEEE Trans. Visualization and Computer Graphics,* vol. 9, no. 2, pp. 115-126, Mar./Apr. 2003.
[28] K.-L. Ma, "Image Graphs—A Novel Interface for Visual Data Exploration," *Proc. IEEE Visualization Conf.,* pp. 81-88, 1999.
[29] 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," *Proc. ACM SIGGRAPH '97,* pp. 389-400, 1997.
[30] M. Mitchell, *An Introduction to Genetic Algorithms.* MIT Press, 1996.
[31] E. Mortensen and W. Barrett, "Intelligent Scissors for Image Composition," *Proc. ACM SIGGRAPH '95,* pp. 191-198, 1995.
[32] H. Pfister, B. Lorensen, C. Bajaj, G. Kindlmann, W. Schroeder, L.S. Avila, K. Martin, R. Machiraju, and J. Lee, "The Transfer Function Bake-Off," *IEEE Computer Graphics and Applications,* vol. 21, no. 3, pp. 16-22, 2001.
[33] M.S. Prieto and A.R. Allen, "A Similarity Metric for Edge Images," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 25, no. 10, pp. 1265-1273, Oct. 2003.
[34] P. Rautek, I. Viola, and M.E. Gröller, "Caricaturistic Visualization," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 5, pp. 1085-1092, Sept./Oct. 2006.
[35] P. Rheingans and D. Ebert, "Volume Illustration: Nonphotorealistic Rendering of Volume Models," *IEEE Trans. Visualization and Computer Graphics,* vol. 7, pp. 253-264, 2001.
[36] W.S. Robinson, *Understanding Phenomenal Consciousness.* Cambridge Univ. Press, 2004.
[37] K. Sims, "Artificial Evolution for Computer Graphics," *Proc. ACM SIGGRAPH '91,* pp. 319-328, 1991.
[38] S. Takahashi, I. Fujishiro, Y. Takeshima, and T. Nishita, "A Feature-Driven Approach to Locating Optimal Viewpoints for Volume Visualization," *Proc. IEEE Visualization Conf.,* pp. 495-502, 2005.
[39] M. Tory, S. Potts, and T. Möller, "A Parallel Coordinates Style Interface for Exploratory Volume Visualization," *IEEE Trans. Visualization and Computer Graphics,* vol. 11, no. 1, pp. 71-80, Jan./Feb. 2005.
[40] F.-Y. Tzeng, E.B. Lum, and K.-L. Ma, "An Intelligent System Approach to Higher-Dimensional Classification of Volume Data," *IEEE Trans. Visualization and Computer Graphics,* vol. 11, no. 3, pp. 273-284, May/June 2005.
[41] I. Viola, M. Feixas, M. Sbert, and M.E. Gröller, "Importance-Driven Focus of Attention," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 5, pp. 933-940, Sept./Oct. 2006.
[42] I. Viola, A. Kanitsar, and M.E. Gröller, "Importance-Driven Feature Enhancement in Volume Visualization," *IEEE Trans. Visualization and Computer Graphics,* vol. 11, no. 4, pp. 408-418, July/Aug. 2005.
[43] P. Viola and W.M. Wells III, "Alignment by Maximization of Mutual Information," *Int'l J. Computer Vision,* vol. 24, pp. 137-154, 1997.
[44] P. Šereda, A.V. Bartrol, I.W. Serlie, and F.A. Gerritsen, "Visualization of Boundaries in Volumetric Data Sets Using LH Histograms," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 2, pp. 208-218, Mar./Apr. 2006.
[45] L. Wang, Y. Zhao, K. Mueller, and A.E. Kaufman, "The Magic Volume Lens: An Interactive Focus+Context Technique for Volume Rendering," *Proc. IEEE Visualization Conf.,* pp. 367-374, 2005.
[46] Z. Wang, A.C. Bovik, and L. Lu, "Why Is Image Quality Assessment So Difficult," *Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing,* pp. 3313-3316, 2002.
[47] D. Weiskopf, "On the Role of Color in the Perception of Motion in Animated Visualizations," *Proc. IEEE Visualization Conf.,* pp. 305-312, 2004.
[48] D. Weiskopf, K. Engel, and T. Ertl, "Interactive Clipping Techniques for Texture-Based Volume Visualization and Volume Shading," *IEEE Trans. Visualization and Computer Graphics,* vol. 9, no. 3, pp. 298-312, May/June 2003.
[49] H.-C. Wong, H. Qu, U.-H. Wong, Z. Tang, and K. Mueller, "A Perceptual Framework for Comparisons of Direct Volume Rendered Images," *Proc. First IEEE Pacific-Rim Symp. Image and Video Technology (PSIVT '06),* pp. 1314-1323, 2006.
[50] M.-L. Wong, T.-T. Wong, and K.-L. Fok, "Parallel Evolutionary Algorithms on Graphics Processing Unit," *Proc. IEEE Congress on Evolutionary Computation,* pp. 2286-2293, 2005.
[51] Y. Wu, H. Qu, H. Zhou, and M.-Y. Chan, "Focus + Context Visualization with Animation," *Proc. First IEEE Pacific-Rim Symp. Image and Video Technology (PSIVT '06),* pp. 1293-1302, 2006.
[52] Y. Wu, H. Qu, H. Zhou, and M.-Y. Chan, "Fusing Multiple Features in Direct Volume Rendered Images," *Proc. Second Int'l Symp. Visual Computing (ISVC '06),* pp. 273-282, 2006.
[53] H. Zhou, M. Chen, and M.F. Webster, "Comparative Evaluation of Visualization and Experimental Results Using Image Comparison Metrics," *Proc. IEEE Visualization Conf.,* pp. 315-322, 2002.

Yingcai Wu received the BEng degree in computer science and technology from the South China University of Technology, China, in 2004. He is a PhD candidate in the Department of Computer Science and Engineering at the Hong Kong University of Science and Technology (HKUST). His research interests are in medical volume visualization and information visualization. He is a student member of the IEEE and the IEEE Computer Society.

Huamin Qu received the BS degree in mathematics from Xi'an Jiaotong University, China, in 1988 and the MS and PhD degrees in computer science from Stony Brook University (State University of New York at Stony Brook), in 2000 and 2004, respectively. He is an assistant professor in the Department of Computer Science and Engineering at the Hong Kong University of Science and Technology. His research interests include medical visualization, information visualization, and mobile graphics. He is a member of the IEEE and the IEEE Computer Society.